UNIVERSIDADE FEDERAL DE MINAS GERAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM
ENGENHARIA MECÂNICA

"DESENVOLVIMENTO DE UM PROGRAMA DIDÁTICO COMPUTACIONAL DESTINADO À GERAÇÃO DE CÓDIGOS DE COMANDO NUMÉRICO A PARTIR DE MODELOS 3D OBTIDOS EM PLATAFORMA CAD CONSIDERANDO A TÉCNICA PROTOTIPAGEM RÁPIDA"

RAFAEL JUAN COSTA DE MIRANDA

Belo Horizonte, 26 de fevereiro de 2009

Rafael Juan Costa de Miranda

“DESENVOLVIMENTO DE UM PROGRAMA DIDÁTICO COMPUTACIONAL DESTINADO À GERAÇÃO DE CÓDIGOS DE COMANDO NUMÉRICO A PARTIR DE MODELOS 3D OBTIDOS EM PLATAFORMA CAD CONSIDERANDO A TÉCNICA PROTOTIPAGEM RÁPIDA”

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Mecânica da Universidade Federal de Minas Gerais, como requisito parcial à obtenção do título de Mestre em Engenharia Mecânica.

Área de concentração: Projeto Mecânico

Orientador: Prof. Antônio Eustáquio de Melo Pertence

Universidade Federal de Minas Gerais

Belo Horizonte

Escola de Engenharia da UFMG

2009

Universidade Federal de Minas Gerais
Programa de Pós-Graduação em Engenharia Mecânica
Av. Antônio Carlos, 6627 - Pampulha - 31.270-901 - Belo Horizonte – MG
Tel.: +55 31 3499-5145 - Fax.: +55 31 3443-3783
www.demec.ufmg.br - E-mail: cpgmec@demec.ufmg.br

"DESENVOLVIMENTO DE UM PROGRAMA DIDÁTICO COMPUTACIONAL DESTINADO À GERAÇÃO DE CÓDIGOS DE COMANDO NUMÉRICO A PARTIR DE MODELOS 3D OBTIDOS EM PLATAFORMA CAD CONSIDERANDO A TÉCNICA PROTOTIPAGEM RÁPIDA"

RAFAEL JUAN COSTA DE MIRANDA

Dissertação/ defendida e aprovada em 26 de fevereiro de 2009, pela Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Mecânica da Universidade Federal de Minas Gerais, como parte dos requisitos necessários à obtenção do título de "Mestre em Engenharia Mecânica", na área de concentração de "Projeto Mecânico".

Prof. Dr. Antônio Eustáquio de Melo Pertence – Universidade Federal de Minas Gerais
Orientador

Prof. Dr. Alexandre Mendes Abrão– Universidade Federal de Minas Gerais
Examinador

Prof. Dr. Alexandre Carlos Eduardo– Universidade Federal de Minas Gerais
Examinador

Prof. Dr. Danilo Amaral– Universidade Federal de Minas Gerais – Suplente

---

Prof. Dr. Haroldo Béria Campos– Universidade Federal de Minas Geral - Suplente

AGRADECIMENTOS

A Deus, por eu ter perseverado e conseguido superar as inúmeras dificuldades que surgiram em meu caminho; por tornar possível mais esta conquista em minha vida.
Aos meus pais por todo carinho, pelos conselhos , pela preocupação com meu presente e futuro, pelo investimento em minha educação, enfim, por todos os valores que acrescentaram em mim.
Ao professor Prof. Dr.Antônio Eustáquio de Melo Pertence, meu orientador; a quem considero um exemplo de professor e de pessoa; pela confiança depositada em mim e o apoio durante toda minha trajetória no Mestrado.
Aos examinadores da banca por participarem na melhoria de meu trabalho ao se disporem a avaliar a dissertação.
Aos amigos e a família, pelos momentos agradáveis de descontração,pelo carinho e atenção e pelo interesse no andamento do mestrado; que foram importantes para recuperar o ânimo e continuar as atividades.

Meus sinceros agradecimentos.

“A mente que se abre a uma nova idéia jamais voltará ao seu tamanho original.”
Albert Einstein

“A genialidade é feita de 1% inspiração e 99% transpiração.”
Thomas Edison

“Vós, porém, não sereis chamados mestres, porque um só é vosso Mestre, e vós todos sois irmãos.”
Mateus 23:8

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA ABREVIATURAS E SIGLAS

ADO *ActiveX Data Objects* (Objetos de dados *ActiveX*)

ANSI *American National Standards Institut*

ASP *Active Server Pages*

BASIC *Beginner's All-purpose Symbolic Instruction Code* (Código simbólico de instruções para iniciantes)

CAD *Computer Aided Design* (Projeto auxiliado por computador)

CAM *Computer Aided Manufacturing* (Manufatura auxiliada por computador)

CPU Central de processamento única

CNC Comando Numérico Computadorizado

CND Comando Numérico Direto

DAO *Data Access Object* (Acesso a dados do objeto)

EIA *Electronic Industries Association* (Associação da indústria de eletrônica)

FDM *Fused Deposition Modeling*(Modelagem por deposição de fundido)

FPM *FreeForm Powder Molding*

GUI *Graphical User Interface* (Interface gráfica com o usuário)

HPLG *Hewlett-Packard Graphics Language* (linguagem gráfica da *Hewlett-Packard*)

IDE *Integrated Development Environment* (Ambiente de desenvolvimento integrado)

IGES *Initial Graphics Exchange Specification* (Especificações gráficas iniciais para troca)

ISO *International Standarization Organization*

MIT *Massachusetts Institute of Technology*

NBS *National Bureau of Standards*

NIST *National Institute of Standards and Technology*

NURBS *Nonuniform rational B-spline curve* (Curva racional não-uniforme *B-spline*)

ODBC *Open Data Base Connectivity* (Conectividade aberta para base de dados)

PC *Personal Compute* (Computador pessoal)

RDO *Remote Data Object* (Objeto de dados remotos)

RP *Rapid Prototyping*(Prototipagem rápida, prototipagem por adição)

STEP *Standard for the Exchange of Product Model Data* (Padrão para troca de dados de modelo)

VBA *Visual Basic for Applications* (*Visual Basic* para aplicações)

2D	Bidimensional, dotado de duas dimensões ou coordenadas
3D	Tridimensional, dotado de três dimensões ou coordenadas

RESUMO

Neste trabalho é apresentado um programa desenvolvido para fins didáticos denominado de *AutoCAM*. Este promove a automação da geração do código G padronizado para máquinas CNC (comando numérico computadorizado), retirando informações de modelos virtuais 3D criados em plataforma *CAD*. O programa desenvolvido na linguagem de programação *Visual Basic*, é capaz de realizar o controle externo de um programa *CAD*, o *AutoCAD*® versão 2008 por intermédio de uma tecnologia de automação do tipo *ActiveX* denominada *VBA (Visual Basic* para aplicações*)* .O programa tem como entradas o modelo tridimensional virtual de uma peça e informações fornecidas pelo usuário por meio da interface do programa; sendo gerado como resultado final um código diretamente aplicável em máquinas CNC. O código gerado possui uma estratégia de construção de uma dada peça de modo semelhante ao método de prototipagem rápida. Neste caso a peça é construída considerando-se as diversas camadas nos planos obtidos através do método de fatiamento direto (*direct slicing*) do modelo tridimensional virtual gerado pela plataforma CAD. Para a validação do programa desenvolvido, foram utilizados um programa gratuito de simulação do processo de fabricação denominado *CNCSimulator* e a fabricação do protótipo físico pela máquina CNC utilizando a prototipagem por retirada de material. Consideram-se como produtos deste trabalho o desenvolvimento do programa didático, a sua aplicação, bem como toda a estratégia elaborada para a obtenção do protótipo físico. Todo o trabalho desenvolvido valida a proposta de empregar o programa como uma ferramenta no ensino de disciplinas relacionadas ao projeto mecânico e processos de fabricação dentro do âmbito da área de Engenharia. A utilização do programa *AutoCAM*, requer do usuário o desenvolvimento de estratégias que possibilitem utilizar quantos modelos 3D forem necessários para representar as geometrias da peça considerando-se os contornos externos e as cavidades, de tal forma que a somatória destas ações se traduza no protótipo final.

Palavras Chave: *Código G, Comando Numérico Computadorizado, fatiamento direto, Manufatura auxiliada por computador, Projeto auxiliado por computador,Prototipagem Rápida, Programa Didático.*

1. INTRODUÇÃO

Os protótipos constituem ferramentas de estudo amplamente utilizadas em projeto. Estes fornecem uma realimentação que influenciará nas decisões e isto afetará etapas futuras de um projeto. Apresentam benefícios já consagrados como a redução dos custos por retrabalho e diminuição do tempo necessário até o início da fabricação do produto final. Também podem ser utilizados como ferramentas de captação de recursos para financiamento de um projeto ao serem apresentados a investidores em potencial.

Devido à amplitude tanto do conceito de projeto quanto dos vieses de estudo existentes dentro de um projeto de maneira geral, se faz necessário que os protótipos sejam versáteis, de sorte que consigam atingir seu objetivo. Esta versatilidade é requerida em termos de características desejadas nos protótipos e dos meios disponíveis a serem utilizados para obtê-los.

O projeto mecânico é o objeto de interesse dentro deste contexto, pois este trata do projeto de peças mecânicas. Desta maneira a prototipagem é abordada no sentido de estudar e desenvolver tecnologias aplicadas à fabricação elaboração de protótipos. Este tipo de abordagem abre diversas frentes de pesquisa como utilização de protótipos virtuais, materiais para prototipagem, tecnologias de fabricação, características dimensionais do protótipo, propriedades físicas do protótipo, otimização de parâmetros das máquinas de prototipagem e geração de trajetória de fabricação de protótipos.

O Laboratório de Projetos Mecânicos do Departamento de Engenharia Mecânica, inserido no Programa de Pós-Graduação em Engenharia Mecânica da UFMG, vem desenvolvendo trabalhos na área de prototipagem (PERTENCE, SANTOS, JARDIM; 2001; SANTOS, 2002), procurando utilizar os resultados obtidos como ferramenta no ensino de disciplinas relacionadas ao projeto mecânico e processos de fabricação dentro do âmbito da área de Engenharia. O presente trabalho vem a dar continuidade nesta linha de pesquisa.

O programa computacional *AutoCAM* como um de seus produtos; avança nos estudos relacionados à utilização de protótipos virtuais, geração de trajetória e tecnologia de fabricação. O avanço na utilização de protótipos virtuais se dá na aplicação da metodologia proposta de gerar códigos diretamente executáveis por máquinas de comando numérico que por sua vez corresponde à trajetória percorrida pela ferramenta no ato da fabricação. Também existiu avanço em termos de tecnologia de fabricação, pois juntamente ao programa

computacional, foi elaborada uma estratégia de utilização que leva em consideração o método de fabricação adotado para obter protótipos físicos.

Obviamente não só a aplicação do projeto mecânico é caracterizada neste trabalho mas também do projeto de um *software*. O programa computacional desenvolvido para fins didáticos, promove a automação da geração de código para máquinas CNC (comando numérico computadorizado), o código G; retirando informações de modelos virtuais 3D criados em plataforma CAD. Este programa desenvolvido na linguagem de programação *Visual Basic*, é capaz de realizar o controle externo de um programa CAD, o *AutoCAD® release 2008* por intermédio de uma tecnologia de automação do tipo *ActiveX* denominada *VBA (Visual Basic for Applications)*.

Levando em consideração o que foi apresentado até então do conteúdo deste trabalho, pode ser abstraído que seu objetivo geral é contribuir para o domínio da tecnologia de fabricação de protótipos produzindo peças destinadas ao auxílio do ensino aos estudantes de Engenharia Mecânica em disciplinas relacionadas às áreas de processos de fabricação, desenho mecânico e projeto de máquinas. Também é possível afirmar que podem ser considerados como objetivos específicos, o exercício de conhecimentos de controle de programas CAD através do uso de interfaces gráficas e o desenvolvimento de estratégias para fabricação de peças utilizando a técnica de prototipagem. Em um primeiro momento utilizando o sistema por retirada de material e posteriormente vislumbra-se a aplicação do sistema por adição de material. Esta tecnologia de prototipagem oferece um campo propício para pesquisas, pois embora já sejam comercializados equipamentos para uso industrial, são poucos os equipamentos encontrados para fins didáticos e de baixo custo.

2. REVISÃO BIBLIOGRÁFICA

2.1.Projeto e protótipo

Um protótipo pode ser considerado um objeto (virtual ou físico) de estudo utilizado no processo recursivo de projeto cujo objetivo é propiciar uma realimentação acerca de alguma característica (qualitativa ou quantitativa) que se deseja avaliar antes de tomar alguma decisão com relação ao produto final. Não se trata de uma definição formal, mas sim uma explanação proveniente da observação de sua funcionalidade dentro do contexto no qual este trabalho está inserido.

É vasta a gama de necessidades de estudo, como pode ser visto em SANTOS (2002), a serem supridas por protótipos e para que estes consigam supri-las devem ser executados de maneira que se consiga assegurar a capacidade de se observar ou mensurar determinada característica, que simplesmente se trata uma questão de coerência. Por exemplo, um estudo hidrodinâmico de um rotor não pode ser realizado com um protótipo de papel, bem como o conforto proporcionado por uma poltrona não pode ser analisado por meio um protótipo virtual.

Os exemplos utilizados como argumento permitem que se chegue a seguinte conclusão: tanto o protótipo com suas características, quanto seu processo de fabricação; estão condicionados ao que se deseja obter do protótipo.

Produtos mais complexos, com suas muitas variáveis, são fatalmente subdivididos em projetos menores de seus componentes. Estes componentes como parte de um todo, não obstante, podem possuir vínculos de dependência entre eles. De maneira que um protótipo de um componente pode ser o subsídio de entradas necessárias para iniciar o projeto de outro em uma etapa posterior.

Alterações sofridas em um determinado componentes serão propagadas a todos os outros que sejam dependentes deste, o que torna mais expressiva a presença da recursividade na atividade de projeto.

Segundo GUANGCHUN (2004), é característica da atividade de projeto de novos produtos necessidade de reduzir o tempo de projeto, que são as sucessivas iterações até a concepção do produto final; os custos e ao mesmo tempo primar pelo aumento da qualidade.

À medida que a complexidade aumenta, mais variadas são as necessidades de estudo do produto que vão desde sua viabilidade econômica a características técnicas e estéticas , conforme apontado por SMITH (1999).

Até agora muito foi dito a respeito da interação do protótipo com o projeto, porém não foi definido em que consiste o projeto.

O termo projeto assume várias conotações como pode ser facilmente notado em diversas esferas da sociedade. É comum se deparar com termos como projeto social, projeto urbano, projeto de governo, projeto de vida, projeto orçamentário, dentre muitos outros diversos. Todos são projetos, mas dentro do contexto deste trabalho, será apresentada uma consideração acerca do que seria um projeto, de modo que esta seja capaz de contemplar o projeto novos produtos e projeto de um *software,* ou seja, um programa computacional. Deste modo pode-se considerar o projeto como atividade ou conjunto de atividades que abstraem as necessidades para o usuário de um produto e, tendo estas informações como entrada; objetivam especificar as características do produto, os processos que envolvem sua fabricação, a seqüência desses processos e a interação entre eles.

2.1.1. Classificação dos protótipos

Com relação a sua constituição os protótipos podem ser classificados de virtuais ou físicos. Os virtuais são modelos gráficos tridimensionais, gerados por um *software* CAD. Atualmente a maioria dos *softwares* CAD no mercado é capaz de emular grandezas físicas como massa, força, temperatura, dentre outros utilizados em simulações numéricas tais como o método dos elementos finitos que vem sendo cada vez mais utilizado na avaliação de propriedades físicas, como deformação, tensões mecânicas, escoamento de fluidos e temperatura.

É comum a prática de executar um modelo virtual anteceder à fabricação de um modelo físico. Este fato se deve à necessidade de projetar não só o produto mas a sua fabricação e para tal, as restrições e parâmetros do processo de interesse a ser utilizado na fabricação do protótipo devem ser estudados (LEE,1999). Para auxiliar nesta tarefa, existem os programas CAM. Estes programas acrescem mais automação no desenvolvimento de produtos, atuando em tarefas como: seleção de processos, parâmetros, geometrias praticáveis e geração de trajetória a fim de se executar uma peça física. Porém a sucessão do modelo físico ao virtual não constitui uma regra, pois existe a possibilidade de se constituir um modelo virtual baseados em dados de uma peça física, obtidos através de um sistema de

aquisição de dados. O processo em questão é denominado “engenharia reversa” (GUANGCHUN, 2004). O objetivo dessa prática é reproduzir ou ainda também avalizar alguma característica relativa à geometria obtida pelo trabalho no modelo físico.

Em se tratando do processo e fabricação de protótipos, também conhecido como prototipagem, os processos podem ser definidos como prototipagem por retirada de material e adição de material. Existem também processos que podem ser considerados mistos, por envolverem tanto a tecnologia de retirada quanto deposição de material na fabricação de protótipos físicos.

Como é uma classificação em termos de processo, ela obviamente se restringe aos protótipos físicos, de maneira que pode ser considerada uma subdivisão dentro da classe dos protótipos físicos.

A Figura 2.1 indica uma proposta de classificação para auxiliar o desenvolvimento e o entendimento do assunto aqui abordado.

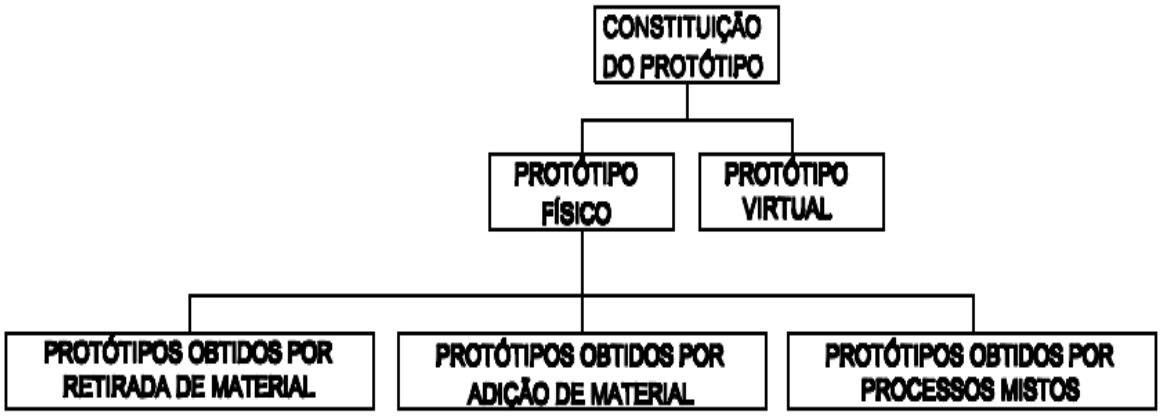


FIG 2.1: Classificação dos protótipos

2.2. Prototipagem por retirada de material

A prototipagem por retirada de material provém da aplicação de processos convencionais de usinagem em máquinas *CNC* na fabricação de protótipos.

A remoção de material pela usinagem de uma peça bruta, confere ao protótipo físico, com boa aproximação; a geometria idealizada no projeto e representada quer seja em desenho bidimensional ou em modelo tridimensional virtual.

A FIG. 2.2 ilustra a fabricação de uma peça utilizando a técnica de prototipagem por retirada de material.

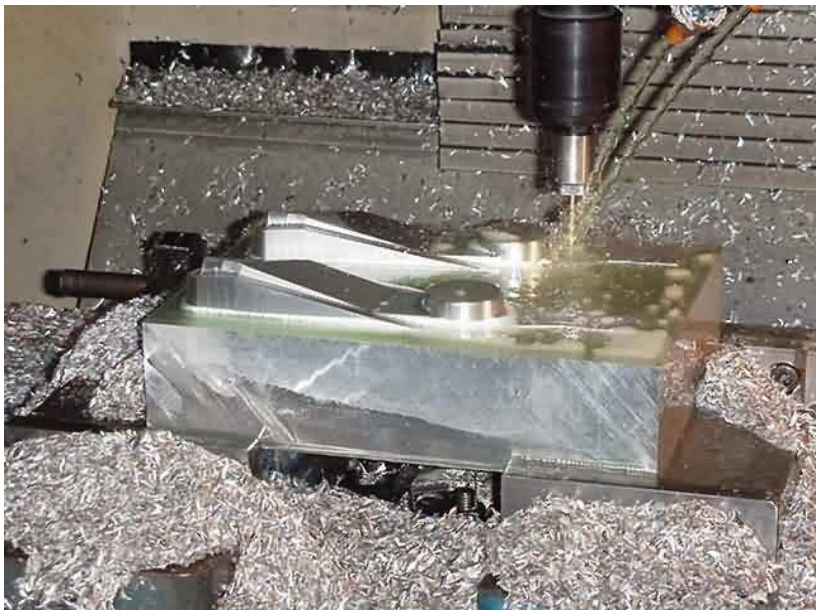


FIG. 2.2: Fabricação de uma peça utilizando a técnica de prototipagem por retirada de material

Embora a prototipagem por retirada (ou remoção) de material não seja definida formalmente como uma tecnologia de fabricação assim como é a prototipagem rápida, sua prática é comum no estudo de processos de fabricação. Os protótipos são utilizados na definição da estratégia de fabricação do produto final, no ajuste de parâmetros para máquinas de usinagem, verificação de desgaste da ferramenta de corte (vida útil), verificação da usinabilidade de materiais, melhoria do acabamento superficial da peça usinada; bem como em outros estudos possíveis (WALSH E CORMIER, 2006).

A fabricação de protótipos por retirada de material não é dependente do CAD como uma fonte de dados tal como depende a prototipagem rápida, embora a vinculação do CAD a usinagem seja bastante comum atualmente. Isto significa que o protótipo físico pode ser obtido sem a necessidade de um modelo virtual.

As máquinas de comando numérico são ferramentas poderosas utilizadas na fabricação de peças usinadas. Estas máquinas podem chegar a ter diferentes tipos de ferramentas que possibilitam realizar diferentes operações de usinagem. A simplicidade de alterar a geometria de uma peça simplesmente alterando seu programa de usinagem aliada a uma grande quantidade de graus de liberdade, constituem uma vantagem muito grande no desenvolvimento de novos produtos, que é o cerne da aplicação dos protótipos.

A precisão dimensional e acabamento superficial das peças usinadas proporcionados pelas máquinas de comando numérico são um diferencial na fabricação de

protótipos, pois as máquinas de prototipagem rápida não se equiparam às máquinas de comando numérico nestes quesitos (VOLPATO,2007).

Os metais e plásticos são materiais comumente empregados na fabricação de peças usinadas tais como os protótipos. A capacidade da prototipagem por retirada de material de trabalhar com os mesmos materiais utilizados no produto final amplia as possibilidades de estudo propiciadas pelos protótipos se for feita uma comparação com os protótipos obtidos por adição de material.

2.3. Prototipagem por adição de material (Prototipagem rápida)

A prototipagem por adição de material, mais conhecida como prototipagem rápida (*Rapid Prototyping ,* RP) consiste na adição de finas camadas ou pequenas partículas, que de algum modo são aglutinadas ,e assim formarem um objeto tridimensional. Este método de fabricação também é conhecido por outros nomes como manufatura por camadas (*layered manufacturing*) ou ainda impressão 3D (*3D printing*) devido ao processo de fabricação dos protótipos em camadas (CHIU E LIAO, 2003).

. Como será contemplado adiante; a prototipagem tem um campo bem abrangente de aplicações, e algumas delas utilizam a capacidade de se obter geometrias de grande complexidade, impraticáveis ou muito complexas para a fabricação por retirada de material. Assim, a prototipagem também fica conhecida também como manufatura de formas livres (*free-form fabrication*). Além da vantagem de trabalhar com geometrias complexas, a prototipagem rápida proporciona uma minimização da intervenção humana no processo de fabricação, consistindo em um acréscimo de automatização no processo de fabricação de um protótipo, reduzindo a possibilidade da ocorrência de erros.

Os diferentes métodos de prototipagem rápida existentes obedecem ao mesmo princípio, o da construção por camadas. O que os diferencia é a maneira como a adição do material é feita, o que também dependerá de qual será o material utilizado para a fabricação de um protótipo.

No campo industrial, um meio bastante competitivo, a prototipagem rápida tem como seu maior benefício até então a possibilidade de reduzir o tempo de desenvolvimento de novos produtos.

A arquitetura se beneficia também da prototipagem da prototipagem pelo fato da redução de tempo proporcionada pelos modelos (GIBSON, KVAN, MING; 2002). Os

modelos (FIG. 2.3 e 2.4) são utilizados para avaliação de características técnicas da execução e viabilidade comercial do projeto, por parte de arquitetos, engenheiros e clientes.

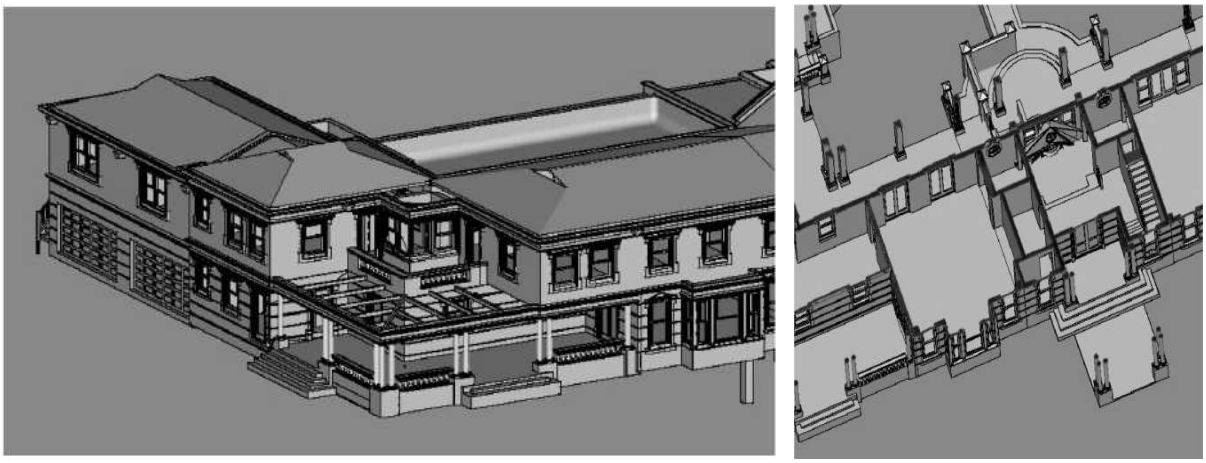


FIG. 2.3 - Modelo virtual de uma edificação. Fonte :DE BEER, BARNARD, BOOYSEN (2004)

FIG. 2.4 – Modelo físico de uma edificação, obtido por prototipagem por adição de material Fonte :DE BEER, BARNARD, BOOYSEN (2004)

Dentre as especialidades da medicina existe uma que se beneficia da prototipagem rápida, utilizando os protótipos em casos de intervenção cirúrgica, que geralmente oferecem muito risco ao paciente. A ortopedia realiza aplicações da prototipagem na fabricação de

protótipos de estruturas ósseas e equipamentos específicos para procedimentos cirúrgicos (SANGHERA, NAIQUE, PAPAHARILAOU; 2001, GOPAKUMAR; 2004).

2.3.1. Método do fatiamento direto

O fatiamento direto (*direct slicing*) conforme JAMIESON E HACKER (1995) ,DOLENC E MÄKELÄ (1996) ,FADEL E KIRSCHMAN (1996),JURRENS (1999), CHANG (2004),KUMAR E CHOUDHURY (2005), CHAKRABORTY E CHOUDHURY (2007), consiste na retirada das informações relativas à trajetória diretamente do modelo em contrapartida à utilização do formato STL, desenvolvido a princípio para estereolitografia (*stereolitography*).

Na técnica do fatiamento direto, tendo como entrada o modelo 3D tem-se a saída sob a forma de camadas 2D fatiadas. Esta técnica vem a resolver problemas existentes na utilização da representação do modelo por malha de superfícies triangulares, na qual o STL é baseado.

Utilizando o fatiamento direto os erros de aproximação são evitados, principalmente nas superfícies curvas. O custo computacional de armazenamento e processamento é menor pois o formato STL é uma estrutura redundante. Este custo computacional acarreta no grande período de tempo necessário para o fatiamento do modelo utilizando o STL. Também como vantagem da aplicação do fatiamento direto tem-se a redução na preparação do modelo para o fatiamento em camadas. No ato da geração das malhas triangulares do STL, não obstante podem ocorrer erros. Então o modelo necessita de uma reparação do para então torná-lo apto ser utilizado.

A utilização do fatiamento direto depende do sistema *CAD* utilizado e requer do sistema do *CAD* sofisticação e capacidade de trabalhar com geometrias espaciais complexas. Segundo JAMIESON,HACKER (1995) existem ainda outras características do fatiamento direto que podem ser consideradas como desvantagens como a impossibilidade de se retornar ao modelo original do qual o fatiado deriva, e portanto da incapacidade até então de alterações no modelo fatiado serem transmitidas ao modelo original. Tal fato não tira seu mérito de ser uma alternativa viável ao formato STL, atualmente o formato mais utilizado na prototipagem rápida.

O programa desenvolvido vem a utilizar também esta técnica de fatiamento direto já aplicada na prototipagem rápida, porém o programa irá utilizar este método na fabricação de protótipos por retirada de material. Na aplicação do fatiamento direto neste trabalho pode

ser dito que haverá uma inversão no sentido de fabricação, pois na prototipagem rápida o protótipo é construído de baixo para cima e na prototipagem por retirada de material, este será fabricado de cima para baixo.

2.4.Usinagem

O papel desempenhado pelos processos de usinagem é o de converter matéria-prima em produto. Os metais são amplamente utilizados como matéria-prima, sejam forjados, fundidos ou laminados. Este fato confere aos processos de usinagem um importante papel na fabricação, pois estes trabalham com materiais que possuem as propriedades físicas desejáveis nos produtos mecânicos projetados, ou seja, os processos de usinagem são aplicáveis tanto na fabricação de um produto finais, seriados ou não; (ASM, 1999) quanto na fabricação de protótipos. Na prototipagem rápida existem pesquisadores que militam na busca por materiais e processos que levem a fabricação de peças finais, cujas propriedades físicas sejam apropriadas e assim conseguir se equiparar à aplicação da usinagem. De acordo com a ASM (1999), os processos de usinagem podem ser agrupados em três categorias: os processos convencionais de formação de cavaco, a usinagem abrasiva e processos não tradicionais.

De maneira genérica, os sete processos convencionais de formação de cavaco são: torneamento, aplainamento, furação, fresamento, corte (por serra), brochamento e a usinagem abrasiva. Estes processos são passíveis de serem realizados por mais de uma máquina-ferramenta e os demais processos existentes derivam destes citados. Todos os processos convencionais, bem como outros derivados deles, são suscetíveis à aplicação do controle numérico (WALSH E CORMIER,2006), porém serão desenvolvidos com mais detalhes s processos de fresamento e o torneamento. Estes são processos mais presentes na fabricação da maioria dos produtos mecânicos devido à versatilidade que eles apresentam.

A usinagem abrasiva constitui uma categoria pela particularidade da ferramenta de corte consistir em uma série de pequenas arestas cortantes das chamadas partículas abrasivas dispostas aleatoriamente, e vindo a constituir a ferramenta conhecida por rebolo. Os demais processos conveincionais possuem ferramentas cujas geometrias e quantidade de arestas cortantes são bem definidas.

Os processos não convencionais por sua vez são processos alternativos que envolvem a formação de cavaco por solicitações de cisalhamento e compressão. Geralmente são processos mais caros que os convencionais. Os processos não convencionais que envolvem aquecimento resultam em distorções na peça acabada e aqueles envolvem trabalho

a frio necessitam de um complemento para reduzir as tensões residuais presentes na peça trabalhada.

2.4.1.Fresamento

O fresamento é a operação de usinagem na qual o metal pode ser removido por uma ferramenta rotativa com múltiplos dentes de corte. Cada dente remove uma porção de material a cada revolução da ferramenta. A complexidade de geometrias possíveis de se realizar com o processo de fresamento está intimamente ligada à natureza dos graus de liberdade existentes entre a ferramenta e a peça.

Existem ferramentas de corte para fresamento com diversas geometrias que são as mais apropriadas ou mesmo específicas para obter uma geometria e acabamento superficial desejados na peça usinada.

Assim como as ferramentas, existem diversas formas construtivas de máquinas, que serão mais apropriadas para a usinagem de uma determinada peça e utilizar uma determinada ferramenta de corte também apropriada como indica a FIG 2.5.

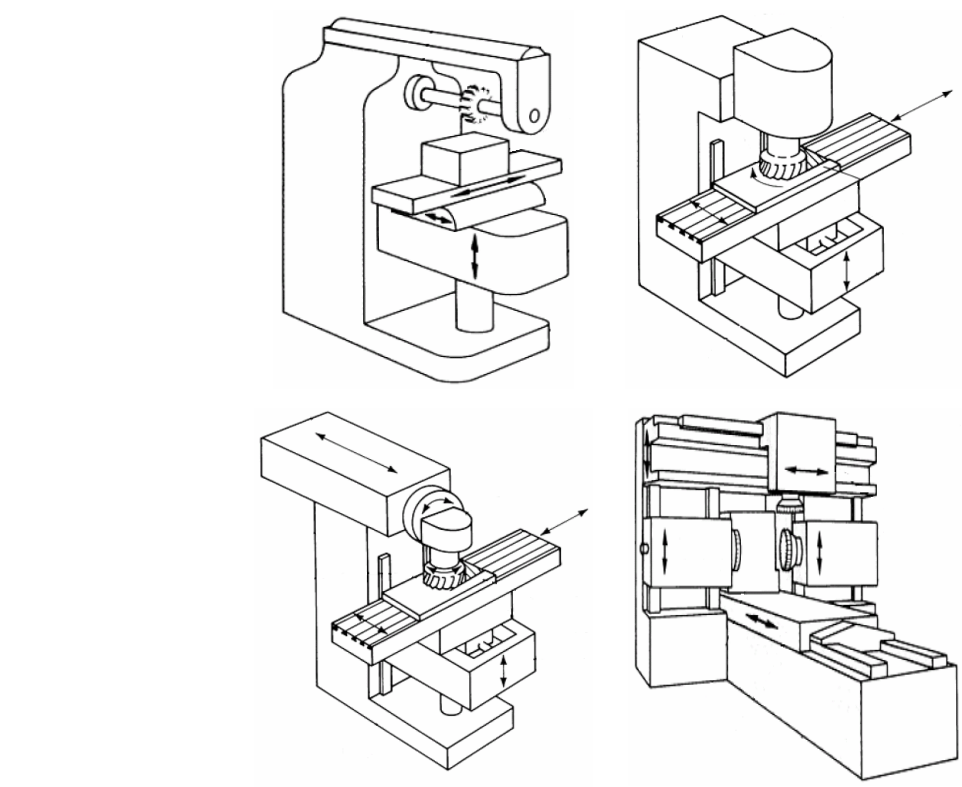


FIG. 2.5- Diferentes formas construtivas de máquinas para usinagem por fresamento e seus respectivos graus de liberdade para a ferramenta de corte e a peça ou material-base.-Adaptação-Fonte (ASM,1999)

As máquinas fresadoras podem ser utilizadas em outros processos de usinagem como por exemplo furação e retífica. A utilização dependerá das características construtivas e dimensões tanto da máquina quanto da ferramenta de corte, de sorte que se obtenha sucesso na adaptação da máquina destinada a princípio ao fresamento, para outro processo. Esta prática de adaptação é realizada na metodologia deste trabalho no ato da fabricação do protótipo físico. Este foi executado através do processo de retífica, adaptando-se uma ferramenta de retífica a uma máquina fresadora de comando numérico.

Não é privilégio das máquinas fresadoras a capacidade de adaptação para um ou mais de um dentre os processos convencionais de usinagem.

Para o escopo deste trabalho não se faz necessária uma pormenorização do fresamento que envolveria assuntos como o mecanismo de da remoção de cavaco, geometria das ferramentas de corte e seus materiais, materiais para fresamento, parâmetros de operação e outros. Como o fresamento não foi empregado na metodologia proposta estas informações pouco acresceriam ao trabalho, porém aos que desejarem pesquisar com maior profundidade e complementar o que aqui foi exposto, podem encontrar em ASM (1999) maiores detalhes sobre a operação de fresamento.

2.5. Sistemas *CAD* e *CAM*

O projeto auxiliado por computador ou CAD é um termo que resume a utilização de sistemas computacionais, *hardware* e *software*; nas tarefas de criação, modificação, análise e otimização de projeto.

A manufatura auxiliada por computador, CAM consiste na utilização de sistemas computacionais nas tarefas de planejamento, gerenciamento e controle da manufatura (fabricação). Quando utilizados em conjunto CAM e CAD colaboram com a integração do projeto com a fabricação tendo como um dos benefícios mais visíveis à redução do tempo decorrido entre o projeto e a fabricação de um produto, como conseqüência de outros benefícios em atividades pertinentes ao projeto e fabricação.

O primeiro sistema CAM foi desenvolvido pelo MIT *(Massachusetts Institute of Technology)* nos EUA, por volta de 1950 para propósitos militares. Posteriormente, na década de 60 do século passado surgem os primeiros sistemas comerciais utilizados em grandes indústrias como a aeroespacial e automobilística.

O MIT também é responsável pelo precursor dos sistemas CAD, desenvolvido em 1960. Até então os projetos eram limitados a desenhos manuais, sujeitos à imprecisão humana. Com a evolução dos computadores, sistemas complexos de *CAD* eram desenvolvidos, mas também como os sistemas CAM*,* eram restritos às grandes empresas. Com o advento dos microcomputadores ou computadores pessoais, os *PCs*; ocorreu uma revolução que acarretou em uma disseminação da utilização do CAD, basicamente pelo baixo custo dos equipamentos, e muitas opções de programas ofertados no mercado.

De maneira gradativa, os sistemas CAD bidimensionais foram sendo substituídos ou mesmo agregados em sistemas mais evoluídos capazes de trabalhar com três dimensões, fato que abriu um campo para um grande desenvolvimento e integração entre CAD e CAM*,* fornecendo mais recursos aos profissionais de projeto e produção.

2.5.1. Armazenamento e transferência de dados em CAD

Uma vez que se tenha construído um modelo virtual, sólido ou de superfície, ou mesmo que seja um desenho bidimensional com o detalhamento da peça; é necessário o armazenamento dos dados em um arquivo. Assim as informações estarão disponíveis para serem consultadas posteriormente, poderão futuramente ser alteradas e disponíveis para outras pessoas envolvidas no projeto, pois podem ser transferidas.

Os formatos dos arquivos CAD são o conjunto formado pelos dados e sua estrutura, utilizados no armazenamento e transferência das informações necessárias para se construir a representação gráfica virtual de um sólido. A estrutura significa o conjunto dados que serão necessários para o programa CAD representar o modelo, quais informações serão agregadas a esse sólido, como: tolerância dimensional, propriedades de densidade, módulo de elasticidade, calor específico, dentre outras; e, finalmente, de que maneira essas informações serão armazenadas e transferidas. O aumento da quantidade de informações inseridas em um modelo implica no aumento de possibilidades de análise, porém com um custo computacional do aumento do tamanho do arquivo.

Embora existam formatos comuns à maioria dos programas CAD, observa-se que, em geral, cada programa CAD possui seu formato de arquivo por uma questão estratégica da empresas de programas CAD. Como existem diversos programas CAD*,* cada qual com seu formato de arquivo, para existir a troca de informações entre programas diferentes é necessário que um programa consiga trabalhar com o formato de arquivo do outro ou que exista um padrão. O padrão não tem o objetivo de substituir o formato particular do programa

CAD, trata-se de um formato neutro para intermediar a troca das informações com outro programa CAD.

O formato IGES *(Initial Graphics Exchange Specification)* foi estabelecido como padrão em 1980 pela NBS (*National Bureau of Standard)*, atualmente conhecida como *National Institute of Standards and Technology* ou NIST. A norma NBSIR 80-1978 foi uma primeira tentativa de se padronizar arquivos para a transferência de dados de modelos virtuais. O projeto foi iniciado no ano anterior em um esforço conjunto de grandes empresas que utilizavam CAD, empresas de programas CAD e o departamento de defesa norte-americano (exército). O IGES ainda existe como padrão, embora ele não tenha sido aprimorado desde o estabelecimento do formato STEP.

O formato STEP,*Standard for the Exchange of Product Model Data*, foi estabelecido como padrão na norma em 1984 pela ISO (*International Standarization Organization)* substituindo o formato IGES.

Dentre os formatos de programas CAD comerciais destacam-se o *Parasolid* e o *ACIS.* O destaque destes formatos se deve ao fato dos programas *CAD* atuais mais utilizados serem derivados do *kernel* de algum desses dois formatos. O *kernel* trata-se de um componente do programa responsável pela administração da interação entre a CPU do computador, sua memória e seus dispositivos de entrada e saída com o aplicativo, no caso o programa CAD.

O QUADRO 2.1 apresenta os programas ou padrões CAD mais utilizados e seus formatos de arquivo.

2.5.2. A Plataforma CAD - *AutoCAD*

O *AutoCAD* teve apresentada sua primeira versão 1.0, criada pela AutoDesk Inc**.**, nos EUA em dezembro de 1982. Pelo fato de sua arquitetura ser aberta, constitui um ambiente propício para o desenvolvimento de aplicativos pelo usuário, permitindo então a utilização em praticamente todas as áreas de desenho e projeto; como engenharia, arquitetura,agrimensura, indústria, científico, design ou qualquer outra aplicação que necessite de desenho de *CAD*. Este desenvolvimento se deu primeiramente através do *Visual Lisp* e migrando para a tecnologia *ActiveX* e o *VBA* (*Visual Basic for Applications*).

Inicialmente, o programa era destinado a desenho mecânico. Devido a sua arquitetura aberta, logo se tornou um padrão para programadores de *sistemas*, contando hoje com milhares de usuários por todo o mundo.

Programa CAD	Empresa de desenvolvimento	Formatos de arquivo	
		De importação	De exportação
Advance Concrete	*GRAITEC*	*DWG, GTC*	*DWG, GTC*
Advance Design	*GRAITEC*	*CIS/2, SDNF, PSS, VRML, DXF, GTC*	*CIS/2, SDNF, PSS, VRML, DXF, GTC*
Advance Steel	*GRAITEC*	*DWG, CIS/2, SDNF, PSS, KISS, GTC*	*DWG, CIS/2, SDNF, PSS, KISS, GTC*
AutoCAD	*Autodesk*	*DXF, DWG, DWS, DWT, WMF, SAT, 3DS, DGN*	*3DS ,DXF, DWG, DWF, DXX, BMP*
Bricscad	*Bricsys*	*DWG, DXF*	*DWG, DXF*
BRL-CAD	*United States Army Research Laboratory*	*DXF, Elysium Neutral Facetted, EUCLID, FASTGEN, IGES, Jack, NASTRAN, Pro/E, STL, TANKILL, Unigraphics, Viewpoint*	*DXF, EUCLID, IGES, Jack, STL, TANKILL, VRML, Obj, X3D*
CATIA	*Dassault Systèmes*	*CGR*	*CGR*
Digital Project	*Gehry Technologies*	*3dxml, IGES, CGR, DWG*	*STL, IGES, STEP*
GStarICAD (baseado no) IntelliCAD)	*Great Star Software*	*DWG, DWF, WMF, DXF, DWT, ACIS, 3D Studio*	*DWF, PDF, WMF, EPS, BMP, EMP, SLD, PNG, SVG, GIF, JPG, TIF, PCX,ACIS*
Autodesk Inventor	*Autodesk*	*ACIS SAT, DXF & DWG, IGES, Pro/Engineer,STEP*	*ACIS SAT, DXF & DWG, IGES, STEP*
CADKEY KeyCreator	*Kubotek*	*ACIS SAT, Catia v4 & v5, DXF & DWG, IGES, Inventor, Parasolid, Pro/Engineer, Solidworks, STEP, UniGraphics*	*ACIS SAT, Catia v4 & v5, DXF & DWG, IGES, Parasolid, STEP*
MicroStation	*Bentley Systems*	*DGN, DXF, DWG, SketchUp, Rhino, PDF, Revit, IFC, gbXML*	*DGN, DXF, DWG, U3D, ADT, PDF, gbXML*
NX	*Siemens PLM Software*	*JT, Parasolid, STEP, DWG/DXF, ProE, SolidWorks, I-deas, CATIA (V4/V5), STL, IGES*	*JT, Parasolid, STEP, DWG/DXF, ProE, SolidWorks, I-deas, CATIA (V4/V5), STL, IGES*
Pro/ENGINEER	*Parametric Technology Corporation*	*STEP, IGES, DXF, DWG, Parasolid, JT ,ASIC*	*CATIA (V4/V5) ,Unigraphics*
QCad	*RibbonSoft*	*DXF R12, DXF 2000*	*DXF, SVG, PDF*
SagCAD	*SagCAD developers*	*DXF*	*DXF*
Shark LT e Shark, Shark FX	*Punch!*	*Template:3D Studio, Acis SAT, AI, PSD, BMP, Catia, DWG, DXF, EPS, Facet, FACT, GIF, Grid Surf, IGS, JPG, PDF, PICT, PNG, ProE, Punch!, Rhino, Sketchup, Spline, STEP, STL, Text, TIFF, COB, OBJ*	*Template:Acis SAT, AI, BMP, CGM, Catia, DWG 12, 13, 14, 2000, 2004, 2008, EPS, Facet, FACT, IGS, JPG, PICT, RAW, STEP, Text, Viewpoint, VRML, OBJ*
Solid Edge	*Siemens PLM Software*	*IGES, STEP, DXF, JT, ACIS (SAT), ProE, SolidWorks, NX, SDRC, Microstation, Inventor, CATIA (V4/V5), Parasolid, AutoCAD, STL, XML, MDS*	*IGES, STEP, STL, PDF, EMS, JT, XGL, XML, DXF, Parasolid, CATIA (V4/V5), ACIS (SAT), Microstation, AutoCAD*
Solidworks	*SolidWorks Corp.*	*Sldprt,sldasm,slddrw,DXF,DWG,Parasolid,IGES, STEP, ACIS, VDAFS,VRML, STL,Catia,ProE ,Unigraphics , Inventor Part, Solid Edge ,CADKEY*	*Sldprt,sldasm,slddrw,DXF,DWG,Parasolid,IGES, STEP, ACIS, VDAFS,VRML, STL,Catia,ProE ,Unigraphics , Inventor Part, Solid Edge ,CADKEY,eDrawings*
SpaceClaim	*SpaceClaim Corporation*	*Rhino (.3DM), IGES, STEP, ProE, SolidEdge, SolidWorks, Inventor, NX, CATIA (V4/V5), Parasolid, AutoCAD*	*Rhino (.3DM), IGES, STEP, DWG, DXF, XAML, STL, VRML, XPS*
TopSolid	*Missler Software*	*IGES, STEP,*	*IGES, STEP, STL, VRML*
VariCAD	*VariCAD*	*STEP (3D), DWG (2D), DXF (2D) , IGES (2D)*	*STEP (3D), STL (3D), IGES (3D/2D), DWG (2D), DXF (2D)*
VectorWorks	*Nemetschek*	*DWG, DXF, PDF, EPSF, 3DS, SAT, Sketchup, IFC , IGES*	*STL, IGES, DWG, DXF, PDF, 3DS, IGES, SAT, IFC, KML, STL*

QUADRO 2.1- Programas CAD comerciais, suas empresas de desenvolvimento e formatos de armazenamento/ transferência de dados.

Outros sistemas de *CAD* também se firmaram como padrão, como o *Micro-Station* e o *Vector Works*. Sistemas de *CAD* (destinados a projeto e desenho), *CAM* (destinados à manufatura), *GIS* (destinados a geoprocessamento) específicos tem sido criados, destinados a engenharias mecânicas,civis, elétricas,à agrimensura,arquitetura, topografia,

estradas, e modelagem. Alguns exemplos destes programas são: *AutoSurf, AutoArchitect, AutoBuilding*, *Cad Overlay*, *Catia*, *GisPlus*, *EMS*, *Hiteck* e *SolidWorks,* .

Desde o seu lançamento em 1982, o *AutoCAD* tem passado por diversas revisões e alterações, melhorando os recursos do programa. Segue abaixo a relação das versões bem como as mudanças significativas ocorridas a cada atualização:

***AutoCAD* Versão 1.0 (R. 1.0) - Dezembro 1982:** A *Autodesk* lança o *AutoCAD* v.1.0 (R1). A especificação R1 não existia, serve apenas como referência.

***AutoCAD* Versão 1.2 (R. 2.0) - Abril 1983:** Surge o *AutoCAD* v.1.2 (R2).

***AutoCAD* Versão 1.3 (R. 3.0) - Agosto 1983:** A *Autodesk* Lança o *AutoCAD* v.1.3 (R3)

***AutoCAD* Versão 1.4 (R. 4.0) - Outubro 1983:** Foi lançado a *AutoCAD* v.1.4 (R4)

***AutoCAD* Versão 2.0 (R. 5.0) - Outubro 1984:** A *Autodesk* lança o *AutoCAD* v.2.0 (R5).

***AutoCAD* Versão 2.1 (R. 6.0) - Maio 1985:** *Autodesk* Lança o *AutoCAD* v.2.1 (R6). As mudanças no *AutoCAD* se tornaram mais significativas, pois nesta versão, surgiam os comandos como *E-LEV*, *VPOINT* e *HIDE*, permitindo extrusões e visualização da plotagem. A partir desta versão que o *AutoCAD* começa a trabalhar em 3D.

***AutoCAD* Versão 2.5 (R. 7.0) - Junho 1986:** O aumento do espaçamento entre os lançamentos das atualizações é compensado pelas evoluções no programa.

***AutoCAD* Versão 2.6 (R. 8.0) - Abril 1987:** São introduzidos dois novos comandos, o *3DLINE* e *3DFACE*.

***AutoCAD* R. 9.0 - Setembro 1987:** Esta versão é muito semelhante à versão 2.6, porém com alterações em sua interface tornado-o mais interativo.

***AutoCAD* R. 10.0 - Outubro 1988:** O *AutoCAD* se torna mais profissional com a introdução um novo sistema de coordenadas *UCS* (*User Coordinates System*) e novos recursos em 3D comandos como *3DPOLY*, uma série de arcos e retas tridimensionais interconectados; *3DMESH* (malha tridimensional), *RULE-SURF* (superfícies balizadas por curvas arbitrárias), *VIEWPORTS* (vistas projetadas dos sólidos 3D) e outros.

***AutoCAD* R. 11.0 - Outubro 1990:** *AutoCAD* R11 é lançado para *MS-DOS* ou *UNIX*, oferecendo suporte a redes, maior controle das variáveis de dimensionamento. O *AutoCAD* começava a se impor como plataforma.

***AutoCAD* R. 12.0 - Junho 1992:** *AutoCAD* R12 para *MS-DOS* ou *UNIX*, se mostra completo e estável com muitos recursos em 2D, características que o converteram em

um padrão no mercado mundial de CAD confundindo com a própria definição do que é o CAD os recursos em 3D passam a ser um módulo vendido separado como o *3D AME*, a adoção de caixas de diálogo para muitas funções que antes era somente pela linha de comando, tornou muito mais fácil de trabalhar

***AutoCAD* R. 12.1 - Novembro 1993:** Esta é a primeira versão do *AutoCAD* R12 para o sistema operacional Windows . O *AutoCAD* se tornou ainda mais fácil de ser personalizado, apresentando uma caixa flutuante de ícones trazia para fácil acesso os comandos mais usados.

***AutoCAD* R. 12.2 - Novembro 1993:** *AutoCAD* LT *For Windows*, é basicamente o mesmo *AutoCAD* R12, porém mais enxuto, com menos recursos, dirigido para pequenos usuários a procura um programa CAD de baixo custo e que fosse capaz de realizar as tarefas desejadas por esses usuários comuns do programa.

***AutoCAD* R. 13.0 - Dezembro 1994:** Inicialmente a versão R13 era tanto para os sistemas operacionais *MS-DOS*, *Windows* e *UNIX*, mas o que deveria ser um programa polivalente se revelou uma fonte de transtornos aos seus usuários, pois foi o *AutoCAD* que se mostrou mais instável até então; mesmo apresentando grandes avanços como a barra de ferramentas com funções de acesso rápido, funções de *layers* (como por exemplo, ligar/desligar), até então habilitada exclusivamente pela caixa de diálogo ou através da caixa de linhas de comando. Surge também como fruto do avanço a possibilidade de importar figuras nos formatos *.GIF*, *.TIFF* e *.BMP*, o uso de textos completos e não apenas linhas individuais , e fontes *TrueType* incrementaram as opções e corretor ortográfico; foram algumas das inovações.

***AutoCAD* R. 13.1 - Outubro 1995:** É uma versão específica para o Windows 95 que tornou o *AutoCAD* mais estável, porém menos estável que a versão R12.

***AutoCAD* R. 14.0 - Março 1997:** Esta versão já não foi lançada para os sistemas operacionais *DOS* e *UNIX*. Poderoso, oferecendo a estabilidade compatível à versão R12 e as facilidades do R13, fixa como a plataforma mais usada no mundo dominando perto dos 70% do mercado mundial de CAD. Apesar dos recursos 3D evoluírem bastante, a *Autodesk*, começa a direcionar os usuários em 3D para ferramentas mais específicas como o *Autodesk Mechanical Desktop,* já em sua versão 3. Mesmo com essa prática evolução das ferramentas de trabalho em 2D não foi estancada. Os modos de seleção se mostravam simplificados, novos comandos foram introduzidos como por exemplo o *AUTOTRACK* e o sistema de plotagem também foi melhorado.

***AutoCAD* R. 14.1 - Junho 1998:** Surge uma primeira versão do programa em português que não foi bem aceita pelos usuários mais experientes, devido à já serem acostumados com os comandos em inglês e à transcrição algumas vezes errôneas dos comandos em inglês para o português.

***AutoCAD* 2000 (R. 15.0) - Março 1999:** A versão R15 é lançada apresentando poucas vantagens em relação à anterior do *AutoCAD* é que se mostrava tão eficiente quanto a versão atual .As mudanças mais notáveis ocorreram no processo de plotagem, onde os parâmetros passaram a serem gravados no próprio desenho, e o surgimento da opção de se criar quantos leiautes fossem desejados com parâmetros distintos, o comando *DDMODIFY* é substituído pelo comando *PROPERTIES*, muito mais completo, habilitando alterar todos os parâmetros no mesmo local. Surge também o comando *DBCONECT*, permitindo o usuário pesquisar e importar objetos e propriedades como *STYLES*, *BLOCKS* e *LAYERS* dos desenhos sem abri-los. Também é de grande importância a inovação trazida com possibilidade do usuário poder abrir vários arquivos de desenhos simultaneamente.

***AutoCAD* 2000i (R. 15.1) - Julho 2000:** Esta foi uma versão integrada à Internet, para conseguir o máximo desta ferramenta de negócios. A integração do *AutoCAD* ofereceu soluções para diversas áreas como arquitetura, engenharia, construção, comunicações, governos, utilidade pública, topografia e manufatura agilizando a disseminação das informações.

***AutoCAD* 2002 (R. 15.2) - Junho 2001:** Aconteceram poucas inovações. Dentre as significativas podem ser citadas: o novo gerenciamento dos blocos com atributos, a sofisticação do comando *ARRAY* e o aparecimento de novas ferramentas de *LAYER* (propriedade gráfica dos elementos gráficos), texto e atributos.

***AutoCAD* 2003 (R. 15.3) - Outubro 2002:** Esta é uma versão do *AutoCAD* pouco conhecida à qual se atribui o título de versão para um futuro lançamento da versão 2004.

***AutoCAD* 2004 (R. 16.0) - Março 2003:** O *AutoCAD* 2004 é uma remodelagem do *AutoCAD* 2002, oferecendo novas e melhoradas funcionalidades que permitem criar com rapidez, compartilhar com facilidade e administrar com eficiência. A versão 2004 passou a oferecer novas características como ferramentas de produtividade, uma interface modernizada, e gráficos da apresentação para a criação dos dados mais rápidos e produtivos.

***AutoCAD* 2005 (R. 16.1) - Março 2004**: O programa passa a proporcionar suporte a um número ilimitado de Camadas de objetos gráficos em um único arquivo. São introduzidos novos comandos para simplificar as tarefas de criar posicionar e editar os textos e para criar tabelas.

***AutoCAD* 2006 (R. 16.2) - Março 2005:** O *AutoCAD* teve sua interface foi melhorada e ocorreu a inclusão de funcionalidade, permitindo uma migração das personalizações da atual para as versões anteriores do programa . Foi introduzida também ao programa uma calculadora.

***AutoCAD* 2007 (R. 17.0) - Março 2006:** O *AutoCAD* 2007 passa disponibilizar novas formas de trabalhar: uma nova forma de conjugar o desenho 2D com o modelo 3D, uma nova forma de fazer projeções e visualizações.. Aparece a opção que viabiliza a conversão dos arquivos de desenho em formato *.PDF*. Esta versão foi elaborada no sentido de: melhorar a capacidade dos projetistas de criar, navegar e editar um projeto conceptual, apresentar claramente o projeto a um público não-técnico e posteriormente documentar o projeto com facilidade utilizando todas as ferramentas de desenho disponíveis.

***AutoCAD* 2008 (R. 17.1) - Março 2007:** A versão 2008 adiciona recursos para ajudar a tornar as tarefas repetitivas do usuário mais fáceis, automatizando muitas delas. Escalonamento de anotação e controle das camadas através do ponto de vista, foram introduzidos no sentido contribuir para a visibilidade das anotações, enquanto que as melhorias de texto, vários líderes e tabelas aprimoradas ajudam a fornecer um nível de precisão estética e profissionalismo.

***AutoCAD* 2009 (R. 17.2) - Março 2008:** Em Março de 2008, a *Autodesk* lançou o *AutoCAD* 2009. (AUTODESK,2009; FINKELSTEIN,2007)

2.5.3. Comando numérico

O comando numérico é um sistema de interpretação de dados que converte um código abstrato inteligível a nós humanos em instruções inteligíveis às máquinas-ferramenta de comando numérico .

Uma máquina de comando numérico por sua vez, nada mais é do que servo-atuador para o comando numérico. As primeiras máquinas de controle numérico (FIG. 2.6) foram desenvolvidas após o final da Segunda Guerra Mundial. O início do seu desenvolvimento foi devido à necessidade da força aérea dos EUA de padrões mais exatos. Em atendimento a essa solicitação, a empresa *Parsons Works* se propôs a desenvolver um sistema servo-controlado por dados fornecidos por computador . Posteriormente a *Parsons Works* veio a se associar ao MIT para então apresentar seu primeiro sistema de comando numérico em 1952, uma fresadora de 3 eixos (ASM, 1999).

O objetivo do comando numérico era tornar o sistema produtivo mais eficiente, proporcionando às máquinas meios de realizarem suas tarefas com agilidade, precisão e repetibilidade incomparáveis às máquinas convencionais.

Inicialmente os sistemas de comando numérico se apresentavam muito complexos na tarefa de escrever o código o que dificultava sua disseminação na indústria. A programação desses primeiros sistemas de comando numérico era realizada através de cassetes de papel ou cartões perfurados nas interfaces existentes nas próprias máquinas (ASM, 1999). A estrutura dos programas das máquinas-ferramenta era baseada em perfis que escreviam as coordenadas bidimensionais da trajetória de corte, enquanto os programas atuais, em sua maioria, são capazes de trabalhar com coordenadas tridimensionais e possuem interface gráfica bastante desenvolvida. Isto permite o usuário, por meio da interface; conduzir a geração do código numérico de uma maneira interativa como a seleção de faces da peça a serem usinadas, a especificação de parâmetros, preferências e ferramentas de corte.

As primeiras máquinas da década de 1950, tinham grandes dimensões e utilizavam tubos de vácuo no controle dos servo-atuadores. Posteriormente por volta de 1960, com o avanço da eletrônica, foram adicionados elementos como circuitos lógicos, controle digital e circuitos integrados; o que possibilitou que os sistemas ficassem cada vez menores e mais baratos.

FIG. 2.6- Centro de usinagem *CNC* com 16 ferramentas da *Cincinnati Milacron*- Fonte ASM (1999)

Da mudança de paradigma ocorrida em meados de 1970 que consistiu na substituição das unidades de comando numérico por computardores; surgiram o comando numérico computadorizado (CNC) e o comando numérico direto (DNC). Estes por sua vez são responsáveis pelo desenvolvimento das tecnologias CAD e CAM (ASM, 1999). As unidades controladoras embutidas nas máquinas de controle numérico, com os avanços tecnológicos (principalmente o aparecimento do microprocessador), os computadores se tornaram parte de praticamente todos os processos industriais modernos, amplamente

utilizados no controle das máquinas-ferramenta, ocasionando a substituição de elementos de *hardware* dos controladores dessas máquinas por *software;* o que acabou por simplificar o *hardware* (amplificadores de sinais, circuitos transdutores, componentes de interface,etc...) . Uma conseqüência direta da evolução dos computadores foi uma redução dos custos das máquinas-ferramenta e por conseguinte sua popularização na indústria.

O CNC é uma tecnologia de um horizonte mais amplo que o controle numérico. Ele não se restringe apenas à geração de trajetória de ferramentas de corte girantes. Sua aplicação se estende atualmente a praticamente todos os processos de usinagem (convencionais ou não convencionais) e apresenta aplicações também em processos de fabricação por conformação (WALSH E CORMIER, 2006).

O DNC se trata da mesma tecnologia de aplicação de computadores no controle de máquinas-ferramenta. O que difere as tecnologias é o fato de que na tecnologia CNC é utilizado um computador para comandar as funções da máquina e já na tecnologia DNC, o comando das funções de um grupo de máquinas (uma célula de fabricação) é realizado por um computador central. Este computador está conectado a uma memória que contém todos os programas de usinagem necessários e então os transmite para as máquinas.

Para as primeiras máquinas de comando numérico este avanço permitiu a substituição dos cassetes de papel aonde os programas eram gravados e a realização de peças de geometrias mais complexas, cujos programas eram grandes demais para a memória dos controladores e; pelo comando numérico direto; poderiam ser enviados em blocos de instruções para a máquina.

2.5.3.1. Estrutura da programação do comando numérico

O programa de usinagem de uma peça é um arquivo de texto, elaborado seguindo uma determinada sintaxe e formato; que contém instruções de usinagem necessárias para a fabricação da peça (ASM, 1999). O programador é a pessoa responsável pela elaboração do código. A tarefa de programar a usinagem de uma peça exige do programador que o mesmo possua conhecimentos em ferramentas de corte, fluidos de corte, sistemas de fixação, usinabilidade de materiais, bem como outros conhecimentos, o que implica que este profissional tem um alto grau de especialização (WALSH E CORMIER, 2006). Cabe ao programador definir as máquinas-ferramenta, a ferramenta de corte, os parâmetros de usinagem e a seqüência dos processos de fabricação. Atualmente existem sistemas CAD/CAM desenvolvidos para auxiliar o programador na tarefa de programação da usinagem

das peças. A metodologia proposta para a geração do código de comando numérico envolve o desenvolvimento de um programa computacional que auxilia na programação da usinagem. Para tanto, a correta sintaxe deve ser conhecida e bem implementada no referido programa de modo que se obtenha a correta interpretação da trajetória pelas máquinas de comando numérico.

Em se tratando da sintaxe, existem os padrões DIN 66025 , ISO 6983-1:1982 e ANSI *(American National Standards Institut)* /*EIA*(*Electronic Industries Association*) RS-274D para o comando numérico de máquinas, que são os adotados pela grande maioria dos fabricantes de máquinas de comando numérico, embora alguns possuam uma sintaxe própria em seus sistemas (FALCK, 2008).

Atualmente os principais fabricantes de sistemas de controle para máquinas CNC são a *GE Fanuc Automation* (uma *joint venture* da *General Eletrics* com a *Fanuc*), *Siemens*, *Mitsubishi*, e *Heidenhain*.

A unidade básica da programação em comando numérico é o bloco, que é visto na forma impressa como uma linha do texto.

Cada bloco pode conter um ou mais palavras, que consistem em letras que descrevem configurações a serem feitas, ou funções a serem realizadas, seguidos de campos numéricos, que por sua vez, fornecem valores a essa função.

A seguinte ordem é necessária para a construção de um bloco:

A. Um bloco opcional para apagar caracteres;
B. Um número opcional para a linha de comando;
C. Um número qualquer de segmentos, que podem ser palavras ou comentários

O sistema de interpretação de código de alguns dos programas CAM, permite que palavras comecem com uma letra qualquer e em qualquer ordem, menos *N*, uma vez que esta letra denota o número de uma linha e deve estar em primeiro lugar. A execução do bloco será realizada na ordem dos números que acompanham *N*. Um exemplo de código é: ***N0001 G0 X23.65.***

Este bloco é construído por três palavras: *N0001*, *G0* e *X23,65*. A primeira palavra é referente á ordem da linha e portanto, da ordem de execução das mesmas. A segunda linha ordena à máquina de comando numérico mover para um ponto com velocidade de posicionamento (geralmente maior que as velocidades de trabalho). As coordenadas para as quais a ferramenta da máquina deve ir serão ditadas pela terceira palavra.

O interpretador permite espaços e tabulações em qualquer lugar dentro de um bloco de código. O resultado da interpretação de um bloco será o mesmo se existissem espaços brancos ou não. A linha *G0X0,1234Y7*, é equivalente à linha *G0 X 0,1234 Y7*.

Linhas em branco são permitidas também na entrada. Elas são ignoradas, pois o algoritmo do sistema interpretador considera que o caso de entrada é irrelevante.

Qualquer letra pode ser maiúscula ou minúscula ou inferior, sem mudar o significado de uma linha.

Existem certas limitações sobre o número ou tipo de palavras que podem aparecer dentro do mesmo bloco. As regras são as seguintes:

- Uma linha pode ter de nenhuma a quatro palavras *G*
- Duas palavras *G* do mesmo grupo não devem aparecer na mesma linha.
- Uma linha deve ter de nenhuma a quatro palavras *M*
- Não podem aparecer na mesma linha
- Para outras letras permitidas, uma linha deve ter apenas uma palavra que comece com uma letra.

Quando um bloco de códigos tem uma ordem específica de execução, ela deve ser considerada como um único comando. Todas as palavras contidas no bloco combinam para produzir uma única seqüência de ações. Estas ações são diferentes do que se as palavras fossem divididas em blocos diferentes. O exemplo a seguir ilustra a questão:

Ex:

n1 x6 – Move da posição atual até a posição "x6".
n2 y3 - Move da posição atual "x6" até a posição "y3".
n3 z2 - Move da posição "z" atual até a posição"z2" moves , mantendo " x6" e "y3".
n10 x6 y3 z2 – Move em conjunto as coordenadas "x", "y" e "z" das posições atuais para "x6", "y3" e " z2".

Embora a posição final resultante do conjunto de blocos *n1* até *n3* e o bloco *n10* seja a mesma, a seqüência dos movimentos se mostra diferente.

Uma vez que toda palavra em comando numérico é composta por uma letra e um número, o significado do valor deste número para o algoritmo de interpretação de comando numérico, deve ser analisado. Um valor real é uma coleção de caracteres que podem ser processados para formar um número. Um valor real deve ser um número explícito, um valor de parâmetro, uma expressão ou um valor unário de operação. Desta maneira um número consiste em uma seqüência de caracteres que representam seu valor que podem ser precedidos

por um sinal opcional positivo ”+” ou negativo “-“, que apresentam uma única separação decimal representada por um ponto ”.”; e que possuem quantos dígitos de precisão forem necessários, que deve ser no mínimo um dígito.Existem dois tipos de números, os inteiros e os decimais. O inteiro não possui a separação das casas decimais feita por um ponto como possuem os decimais.

Uma palavra, na programação em comando numérico, como já foi delineado; consiste em uma palavra reconhecia sucedida de um valor real. O QUADRO 2.2 a seguir mostra as palavras reconhecidas pelos algoritmos de interpretação do comando numérico bem como seus significados.

De acordo com o significado e portanto da função das palavras, esta podem ser divididas em grupos.

O primeiro grupo de palavras a ser introduzido é o grupo de palavras de “número de linha”. São as letras “N” já mencionadas anteriormente seguidas por números inteiros entre 0 e 99999. Números de linha não são checados exceto pela quantidade de dígitos. Não é necessário numerá-las pois elas não serão utilizadas pelo programa, mas podem ser um artifício para o programador verificar o programa.

Palavra do comando numérico	**Significado**
D	Compensação de raio da ferramenta
F	Avanço de corte
G	Função geral
H	Comprimento de deslocamento (*offset*) da ferramenta
I	Distanciamento do eixo “X” para arcos
J	Distanciamento do eixo “Y” para arcos
K	Distanciamento do eixo “Z” para arcos
L	Número de repetição de ciclos
M	Funções de miscelânea
N	Número da linha
P	Alongar tempo (em ciclos)
Q	Chave usada com a palavra G10
R	Raio de um arco
S	Plano de ciclo
T	Escolha de ferramenta
X	Eixo “X” da máquina
Y	Eixo “Y” da máquina
Z	Eixo “Z” da máquina

QUADRO 2.2- Palavras do vocabulário do comando numérico

Dando prosseguimento, serão tratadas as palavras “de eixo”. Em um exemplo anterior este tipo de palavra já foi ilustrado. Uma palavra *X* indica o eixo *X*. A palavra *X 30*, significa que a máquina-ferramenta deverá mover no eixo *X*, 30 unidades, que são normalmente milímetros. Assim também ocorre para os eixos *Y* e *Z*. Existem outras palavras que são relacionadas à rotação em torno desses eixos e o deslocamento em relação a eles, uma vez que *X*, *Y*, e *Z*, correspondem a translações, porém nem todas as máquinas reconhecem estes comandos pois é necessário que as máquinas possuam mais de três eixos.

As palavras *A*, *B* e *C*, correspondem à rotação em trono dos eixos *X*,*Y*, e *Z* respectivamente. Já as palavras *U*, *V*, e *W* são relacionadas a deslocamentos paralelos aos eixos *X*,*Y*, e *Z* respectivamente.

As palavras preparatórias são parte das palavras de funções gerais (QUADRO 2.2) que alteram o estado da máquina. Algumas mudam o movimento linear para interpolação circular (arcos e círculos) e outras podem mudar as unidades de medidas de milímetros para polegadas.

O fato é que a maioria das palavras *G* está relacionada a movimentos ou um conjunto de movimentos. No QUADRO 2.3, são enumeradas as palavras do grupo de preparação, ou palavras preparatórias.

Para finalizar os grupos das palavras, serão tratadas adiante as palavras de miscelâneas, as palavras *M*.

As palavras *M* são usadas para controlar o grande número funções de entrada e saída (*I/O*) de uma máquina-ferramenta . Palavras *M* podem iniciar a rotação e a injeção de fluido de corte; também podem sinalizar o final de um programa ou uma parada no âmbito de um programa.

A lista completa palavras *M* disponíveis no padrão RS274NGC está no QUADRO 2.4.

Muitos códigos *G* e *M* causam à máquina mudança de um modo para o outro modo que permanece ativo até algum outro comando mudar seu modo implícito ou explícito. Esses comandos são chamados de modais.

Por exemplo, a habilitação do fluido refrigerante é um comando modal. Se o refrigerante está ligado, ele permanece ligado até que ele esteja explicitamente desativado.

Os códigos *G* também são modais. Se um comando *G01* ,de interpolação linear, é emitido em uma linha, ele será executado novamente sobre a próxima linha a menos que seja dado um comando especificando um movimento diferente ou algum outro comando que implicitamente anula *G01* seja dado.

G0	Posicionamento rápido	G59	Utilizar sistema 6 predefinido de coordenadas de trabalho
G01	Interpolação linear	G59.1	Utilizar sistema 7 predefinido de coordenadas de trabalho
G02	Interpolação circular e helicoidal (sentido horário)	G59.2	Utilizar sistema 8 predefinido de coordenadas de trabalho
G03	Interpolação circular e helicoidal (sentido anti-horário)	G59.3	Utilizar sistema 9 predefinido de coordenadas de trabalho
G04	Habilitar	G80	cancelar modo de movimento
G10	Definição de origem para o sistema de coordenadas	G81	Furação em ciclo
G17	Seleção no plano xy	G82	Furação com habilitação de ciclo
G18	Seleção no plano xz	G83	Ciclo de furação para quebra de cavaco
G19	Seleção no plano yz	G84	Ciclo de faceamento (mão direita)
G20	Seleção do sistema em polegadas	G85	Alargamento de furos não habilitado, fim do ciclo
G21	Seleção do sistema em milímetros	G86	Alargamento da furação em ciclo, parada da rotação, saída do ciclo
G40	Cancelar compensação de diâmetro da ferramenta de corte	G87	ciclo reverso de Alargamento da furação
G42	Começar compensação de diâmetro da ferramenta de corte	G88	Alargamento da furação em ciclo, parada da rotação, saída manual do ciclo
G43	Deslocamento do comprimento da ferramenta	G89	Sistema de coordenadas absolutas
G49	Cancelar compensação para o comprimento da ferramenta	G90	Sistema de coordenadas incrementais
G53	Movimento da máquina no sistema de coordenadas	G91	Modo de distância incremental
G54	Utilizar sistema 1 predefinido de coordenadas de trabalho	G92	Deslocamento do Sistema de coordenadas
G55	Utilizar sistema 2 predefinido de coordenadas de trabalho	G92.2	Cancelar deslocamento do Sistema de coordenadas
G56	Utilizar sistema 3 predefinido de coordenadas de trabalho	G93	Modo de execução inverse do avanço
G57	Utilizar sistema 4 predefinido de coordenadas de trabalho	G94	Modo de avanço por minuto
G58	Utilizar sistema 5 predefinido de coordenadas de trabalho	G98	Retorno ao nível inicial em ciclos

QUADRO 2.3- Palavras preparatórias do vocabulário do comando numérico

M0	Parada programa	M8	Fluido refrigerante (fluxo líquido) habilitado
M1	Parada opcional no programa	M9	Fluidos refrigerantes habilitados (fluxo líquido e névoa)
M2	Fim do programa	M26	Habilitar fixação automática no eixo b
M3	Rotação da ferramenta em sentido horário	M27	Desabilitar fixação automática no eixo b
M4	Rotação da ferramenta em sentido anti-horário	M30	Fim do programa, movimentação das paletes, e *reset*.
M5	Parar rotação	M48	Habilitar sobreposição de velocidade e avanço
M6	Trova de ferramenta	M49	Desabilitar sobreposição de velocidade e avanço
M7	Fluido refrigerante habilitado (névoa)	M60	Parada do programa e movimentação das paletes

QUADRO 2.4- Palavras de miscelâneas do vocabulário do comando numérico

Os comandos modais estão organizados em conjuntos chamados grupos modais.Apenas um membro do grupo pode ser um modal em vigor em cada momento. Em geral, um grupo modal contém comandos para o qual é logicamente impossível para dois membros de estar em vigor ao mesmo tempo. Sistemas de medidas em polegadas e em milímetros são modais. A máquina-ferramenta pode ser de muitos modos ao mesmo tempo, com um modo de cada grupo deve ser em vigor.

2.6. Programação

2.6.1. A linguagem e ambiente de programação *Visual Basic*

O *Visual Basic* (PERRY,1999) é uma linguagem de programação desenvolvida pela *Microsoft*, baseada na linguagem *BASIC* (*Beginner's All-purpose Symbolic Instruction Code)* criada, pelos professores John George Kemeny e Thomas Eugene Kurtz em *Dartmouth College*, 1964; com fins didáticos. A linguagem objetivava a oferecer aos iniciantes em programação uma alternativa menos complexa do que linguagens existentes como *COBOL*, *FORTRAN* e *Assembler*, que necessitavam de maior conhecimento para desenvolver de modo adequado a atividade de programação.

Como qualquer outra linguagem de programação, o *Visual Basic*, consiste em uma coleção de instruções (e opções de instruções) para o computador, denominadas argumentos.

A estrutura do *Visual Basic* é orientada a eventos, o que significa que trata ocorrências que dão início a alguma rotina de trabalho. Isso implica também que o programa permite o uso de objetos, mas não a sua criação, pois não é uma linguagem orientada a objetos. Como evolução do programa; foi agregado a ele um ambiente de desenvolvimento integrado *IDE* (*Integrated Development Environment*) totalmente gráfico, facilitando a construção da interface das aplicações do tipo *GUI* (*Graphical User Interface*).As aplicações constituem um tipo de interface do usuário que permite a interação com dispositivos digitais através de elementos gráficos como ícones e outros indicadores visuais, em contraste a interface de linha de comando. A interação é feita geralmente através do *mouse* ou teclado, com os quais o usuário é capaz de selecionar símbolos e manipulá-los de forma a obter algum resultado prático.

Em suas primeiras versões, o *Visual Basic* não permitia acesso a bancos de dados, sendo portanto voltado apenas para iniciantes, mas devido ao sucesso entre as empresas que faziam uso de componentes adicionais fabricados por terceiros, para acesso a dados a linguagem adotou tecnologias como *DAO* (*Data Access Object*), um padrão de software

usado em engenharia de software , *RDO* (*Remote Data Object*), que permite a criação de interfaces que podem invocar *ODBC* (*Open Data Base Connectivity*), um padrão para acesso a sistemas gerenciadores de bancos de dados ; e *ADO* (*ActiveX Data Objects*), um mecanismo onde os programas o utilizam para a troca de informações com bancos de dados, também da *Microsoft*.

A adição desses componentes, permitiu rápido e fácil acesso a bases de dados. Mais tarde, foi adicionada também a possibilidade de criação de controles *ActiveX*, e, com a chegada do *Visual Studio .NET*, o *Visual Basic* se tornou uma linguagem totalmente orientada a objetos.

O *ActiveX* é um conjunto de *softwares* criado para facilitar a integração entre diversas aplicações como o *Microsoft Word*, *Microsoft Excel*, *Microsoft Access* e *softwares* CAD como o *AutoCAD*.

Objetos são estruturas que combinam propriedades e métodos. As propriedades são características dos objetos, que podem ser acessadas e/ou alteradas pelo programador tanto em tempo de projeto (quando o projeto está sendo desenvolvido) quanto em tempo de execução (quando o aplicativo está sendo executado). Já os métodos são rotinas internas ao objeto que servem para executar determinadas ações.

O *Visual Basic* é atualmente, parte integrante do pacote de programação *Microsoft* Visual Studio. Sua versão mais recente faz parte do pacote *Visual Studio .NET*,é voltada para aplicações *.Net*. Sua versão anterior fez parte do *Microsoft Visual Studio 6.0*, ainda muito utilizada entre programadores de todo mundo .

Existem várias linguagens derivadas do *BASIC*, das quais podem ser citadas a *VBScript*, *Visual Basic .NET* e *VBA* .

O *VB Script* é a linguagem *default* (por definição) para *ASP* (*Active Server Pages*), uma estrutura de programação que dentre outras atribuições, torna possível executar consultas a banco de dados, através da biblioteca de componentes *ActiveX*) e pode ser usada na programação no sistema operacional Windows e de páginas da Internet.

O *Visual Basic .NET* é a nova versão do *Visual Basic*, que é parte integrante da plataforma Microsoft .NET. Essa versão não é totalmente compatível com as versões anteriores, mas existe a possibilidade de converter códigos antigos, que após uma revisão podem ser usados no *Visual Basic* .NET que usa o paradigma de "orientação ao objeto".

O *VBA* permite a criação de *macros*, e está integrado em todos os produtos da família de produtos *Microsoft Office* como: *Word, Excel, Access, Outlook, PowerPoint* e *FrontPage*; e também em outros produtos tais como os *softwares Visio* e *AutoCAD*.

Macro, em termos de programação, é uma abstração que define como um padrão de entrada deve ser substituído por um padrão de saída de acordo com um conjunto de regras. Desta maneira esta estrutura permite a realização de tarefas repetitivas previamente programadas que serão realizadas de maneira ágil e menos propensa a ocorrência de erros (GREENWALD, MAUREEN, 1959).

No desenvolvimento do programa computacional *AutoCAM,* a programação foi realizada na linguagem *Visual Basic* e foi utilizado o ambiente de programação *AutoCAD VBA*. Assim, o programa computacional concluído existe até então sob a forma de um projeto de *software* que tem sua rotina inicializada por um *Macro* inserido na interface do *AutoCAD.* Desta forma não só a proposta da geração do código numérico se concretiza, mas também o exercício destas tecnologias de automação de *softwares*. Elas permitem aplicações das mais diversas, extrapolando até a linha de pesquisa de prototipagem abordada neste trabalho. Para utilizar estas tecnologias são necessários os conhecimentos de sua estrutura, sintaxe e recursos, para então estar capacitado a criar aplicações utilizando a automação de *softwares*.

2.6.2. Elementos de programação no *Visual Basic*

Um elemento de programação é um elemento declarado que é definido em uma declaração de estrutura. Elementos declarados incluem variáveis, constantes, enumerações, classes, estruturas, módulos, interfaces, procedimentos, parâmetros de procedimento, retornos de funções, referências a procedimentos externos, operadores, propriedades, eventos e representantes. As declarações de estruturas são as seguintes:

- Dim declaração
- Declaração *Const*
- Declaração *Enum*
- Declaração *Class*
- Instrução *Structure*
- Declaração de Módulo
- Declaração Interface
- Instrução *Function*
- Instrução sub-rotina
- Instrução *Declare*
- Declaração *Operator*

- Propriedade declaração
- Declaração de evento
- Instrução *Delegate*

Uma característica de um elemento de programação é um aspecto do elemento que afeta como o código pode interagir com ele. Cada elemento de programação possui uma ou mais das seguintes características associadas a ele:

- Tipo de dados : Os valores que o elemento pode conter, e como ele armazena esses valores.
- Vida útil : O período de tempo de execução durante o qual o elemento está disponível para uso.
- Escopo: O conjunto de todo os códigos que podem se referir ao elemento sem qualificar seu nome.
- Nível de acesso : A permissão para o código fazer uso do elemento. Características dos elementos

O QUADRO 2.5 a seguir mostra os elementos declarados e as características que se aplicam a cada um deles.

Elemento	Tipo de dados	Tempo de Vida	Escopo	Nível de Acesso
Variável	Sim	Sim	Sim	Sim
Constante	Sim	Não	Sim	Sim
Enumeração	Sim	Não	Sim	Sim
Estrutura	Não	Não	Sim	Sim
Propriedade	Sim	Sim	Sim	Sim
Método	Não	Sim	Sim	Sim
Procedimento (*Sub* ou *Function*)	Não	Sim	Sim	Sim
Parâmetro de procedimento	Sim	Sim	Sim	Não
Retorno de função	Sim	Sim	Sim	Não
Operador	Sim	Não	Sim	Sim
Interface	Não	Não	Sim	Sim
Classe	Não	Não	Sim	Sim
Evento	Não	Não	Sim	Sim
Delegado	Não	Não	Sim	Sim

QUADRO 2.5- Relação de elementos de programação do *Visual Basic* e suas características

2.6.2.1. Tipo dos dados no *Visual Basic*

Dado de um elemento de programação se refere a que tipo de dados ele pode armazenar e como ele os armazena. Tipos de dados se aplicam a todos os valores que podem ser armazenados na memória do computador ou participar da avaliação de uma expressão. Toda variável,quer seja: “literal”, “constante”, “enumeração”, “propriedade”, “parâmetro de procedimento”, “argumento de procedimento” ou ”valor de retorno”, tem um tipo de dado.

Um elemento de programação é definido com uma declaração, e o programador deve especificar seu tipo de dado com a cláusula “As”.

O QUADRO 2.6, a seguir; mostra declarações que são utilizadas para declarar vários elementos.

Elemento de programação	**Declaração de tipo de dados**
Variável	Dim declaração (*Visual Basic*). ***Ex: Dim amount As Double*** ***Ex: Static yourName As String*** ***Ex: Public billsPaid As Decimal = 0***
Literal	Como um caractere de tipo literal, ***Ex: Dim searchChar As Char = "." C***
Constante	Em um Declaração Const (*Visual Basic*). ***Ex: Const modulus As Single = 4.17825F***
Enumeração	Num Declaração Enum (*Visual Basic*) ***Ex: Public Enum colors***
Propriedade	Em um Propriedade declaração. ***Ex: Property region() As String***
Parâmetro de procedimento	Instrução tipo sub-rotina (*Visual Basic*), Instrução tipo *Function* (*Visual Basic*) ou Declaração tipo *Operator* ***Ex: Sub addSale(ByVal amount As Double)***
Argumento de procedimento	No código de chamada, cada argumento é um elemento de programação que já foi declarado, ou uma expressão contendo elementos declarados. ***Ex: subString = Left(inputString , 5)***
Valor de retorno de Propriedade	Em uma instrução *Function* (*Visual Basic*) ou Declaração tipo *Operator* **Ex: *Function convert(ByVal b As Byte) As String***

QUADRO 2.6- Exemplos de declaração dos elementos de programação na linguagem *Visual Basic*

2.6.2.2. Vida útil dos elementos de programação no *Visual Basic*

A vida útil de um elemento de programação é o período de tempo durante que ele está disponível para uso. Variáveis são apenas elementos que tenham vida útil. Para isso, o compilador trata Parâmetro de Procedimentos e função, retorna como casos especiais de variáveis. O tempo de vida de uma variável representa o período de tempo durante o qual ela pode conter um valor. Seu valor pode mudar durante sua vida útil, mas ela sempre contém algum valor.

Um variável de membro (declarada no nível de módulo, fora de qualquer procedimento) normalmente tem o mesmo tempo de vida como o elemento no qual é declarada. Uma variável declarada em uma classe ou estrutura compartilhada existe como uma cópia separada para cada instância da classe ou estrutura no qual é declarada. Cada variável tem o mesmo tempo de vida como sua instância. No entanto, uma variável compartilhada (*Shared*) tem apenas um único tempo de vida, que dura o tempo todo em que o seu aplicativo estiver sendo executado.

Uma variável local (declarada dentro de um procedimento) existe somente enquanto o procedimento no qual é declarada está sendo executado. Isso se aplica também aos parâmetros do procedimento e para qualquer função de retorno. No entanto, se esse procedimento chama outros procedimentos, as variáveis locais mantêm seus valores durante a execução os procedimentos chamados.

Uma variável local da vida útil começa quando o controle entra o procedimento no qual é declarada. Cada variável local é inicializada para o valor padrão para seu tipo de dados assim o procedimento começa em execução. Quando o procedimento encontra uma instrução *Dim* que especifica valores iniciais, ele define essas variáveis para esses valores, mesmo se o código já atribuiu outros valores a eles.

Cada membro de uma variável de estrutura é inicializado como se fosse uma variável separada. Da mesma forma, cada elemento de uma variável de matriz é inicializado individualmente.

Variáveis declaradas dentro de um bloco em um procedimento (como um *loop For*) são inicializadas na entrada para o procedimento. Esses inicializações terão efeito quer seu código já executa o bloco, ou não.

Quando um procedimento termina, os valores das suas variáveis locais não são preservados e o *Visual Basic* recupera sua memória. Na próxima vez em que o procedimento for chamado, todas as suas variáveis locais são criadas e reinicializadas.

Quando uma instância de uma classe ou estrutura termina, suas variáveis compartilhadas perder sua memória e seus valores. Cada nova instância da classe ou estrutura cria e reinicializa suas variáveis compartilhadas. No entanto, variáveis compartilhadas (*Shared*) são preservadas até que seu aplicativo interrompe a execução.

Quando se declara uma variável local com a palavra-chave estática (*Static*), sua vida útil é maior que o tempo de execução do seu procedimento. O QUADRO 2.7 mostra como a declaração procedimento determina como longo uma variável estática existe.

2.6.2.3. Escopo dos elementos de programação no *Visual Basic*

O escopo de um elemento de programação é o conjunto de todo o código que pode referir-se a ele sem qualificar. Um elemento pode ter escopo em um dos seguintes níveis descritos no QUADRO 2.8.

Procedimento local e o compartilhamento	Vida útil da variável estática começa	Vida útil da variável estática extremidades
Em um módulo (compartilhado por padrão).	Na primeira vez o procedimento é chamado	Quando o aplicativo terminar a execução
Em uma classe ou estrutura, compartilhada (procedimento é não é um membro de instância).	Na primeira vez que o procedimento é chamado em uma instância específica ou na classe ou estrutura nome próprio	Quando o aplicativo terminar a execução
Em uma instância de uma classe ou estrutura, não compartilhada (procedimento é um membro de instância)	Na primeira vez o procedimento é chamado na instância específica	Quando a instância é lançada para coleta de lixo (GC)

QUADRO 2.7- Estrutura dos dados na linguagem *Visual Basic*

Nível de escopo	Descrição
Escopo de bloco	Disponível apenas dentro do bloco de código no qual é declarada
Escopo de Procedimento	Disponível para todo o código dentro do procedimento no qual é declarado
Escopo de Módulo	Disponível para todo o código dentro do módulo, classe ou estrutura no qual é declarado

QUADRO 2.8- Apresentação do escopo dos elementos de programação na linguagem *Visual Basic* e descrição do escopo

Esses níveis de escopo aumentam a partir de menor para o maior, onde escopo mais restrito significa que um menor conjunto de código pode se referir ao elemento sem qualificação.

Quando se especifica o escopo de um elemento ao declará-lo. O escopo pode depender os seguintes fatores:

- A região (bloco, procedimento, módulo, de classe ou estrutura) na qual se declara o elemento
- O nível de acesso que se declara para o elemento

Um elemento de programação está disponível em toda a região na qual é declarado e todo o código na mesma região pode referir-se ao elemento sem qualificar seu nome. Um bloco é um conjunto de instruções envolto em instruções de declaração de ínicio e término , como os seguintes:

- *Do* e *Loop*
- *For[Each]* e *Next.*
- *If* e *End If*
- *Select* e *End Select*
- *SyncLock* e *End SyncLock*
- *Try* e *End Try*
- *While* e *End While*
- *With* e *End With*

Se uma variável é declarada em um bloco, ela pode ser usada apenas dentro desse bloco. No exemplo a seguir, o escopo da variável inteira *cube* é o bloco entre *If* e *End If*,e pode não mais ser referir a *cube* quando a execução passa para fora do bloco.

Ex:

```
If n < 1291 Then
  Dim cube As Integer
  cube = n ^ 3
End If
```

Um elemento de programação declarado dentro de um procedimento não está disponível fora desse procedimento. Somente o procedimento que contém a declaração pode usá-lo. Variáveis nesse nível são conhecidas como variáveis locais.

Escopos de procedimento e bloco estão intimamente relacionados. Se uma variável for declarada dentro de um procedimento, mas fora de qualquer bloco dentro desse procedimento, pode ser considerado que a variável tivesse escopo de bloco, onde o bloco é todo o procedimento.

Por conveniência, o termo “simples nível” de módulo aplica-se igualmente a módulos, classes e estruturas. Elementos nesse nível podem ser declarados colocando a instrução de declaração fora de qualquer procedimento ou bloco mas dentro do módulo, classe ou estrutura.

Quando se faz uma declaração ao nível de módulo, o nível de acesso escolhido determina o escopo.

Os elementos para os quais se declara o nível de acesso privado (*Private*) estão disponíveis para cada procedimento desse módulo, mas não para qualquer código em um módulo diferente. A instrução *Dim* em nível de módulo tem por padrão privado se não for utilizada nenhuma palavra-chave de acesso de nível. No entanto, pode ser feito o nível de escopo e o acesso mais óbvio usando a palavra-chave "privado" (*Private*) na instrução *Dim*.

As variáveis locais têm melhor aplicabilidade para qualquer tipo de cálculo temporário. Nomes de variáveis locais são não suscetíveis a entrar em conflito. Por exemplo, vários procedimentos diferentes podem ser criados contendo uma variável chamada *intTemp* desde que cada *intTemp* seja declarada como um variável local. Assim, cada procedimento reconhece apenas sua própria versão do *intTemp*. Qualquer procedimento pode alterar o valor em seu *intTemp* local sem afetar as variáveis *intTemp* nos outros procedimentos.

As variáveis locais consomem memória somente enquanto o procedimento estiver sendo executado. Sua memória é liberada quando o procedimento retorna para o código de chamada.

Por contraste, variáveis compartilhadas (*Shared*) e estáticas (*Static*) consomem recursos de memória até que seu aplicativo é interrompido, portanto, devem ser utilizadas somente quando necessário. Variáveis de Instância consomem memória enquanto sua instância continua a existir, o que as torna menos eficiente do que as variáveis locais, mas possivelmente mais eficiente do que variáveis compartilhadas ou estáticas.

Em geral, quando se declara qualquer variável ou constante, é boa prática de programação tornar o escopo tão restrito quanto possível, visando conservar a memória e minimiza as chances do código erroneamente referir-se a variável errada.

2.6.2.4. Nível de acesso dos elementos de programação no *Visual Basic*

O nível de acesso dos elementos de programação é a extensão da capacidade de ser acessar o elemento.Isto é determinado não apenas por como o elemento é declarado em si, mas também pelo nível de acesso do recipiente do elemento. O código que não é possível acessar um elemento contendo não pode acessar qualquer de seus elementos contidos, mesmo aqueles declaradas como pública (*Public*).

A palavra-chave "pública" na instrução de declaração, especifica que os elementos podem ser acessados a partir de código em qualquer lugar no mesmo projeto, de outros projetos que fazem referência do projeto, e do qualquer conjunto de módulos (*assembly*) criados a partir do projeto.

Pode se empregar *Public* somente em módulos ou interfaces. Isso significa que você pode declarar um elemento público no nível de um arquivo de origem ou espaço para nome, ou dentro de uma interface, módulo, de classe ou estrutura, mas não em um procedimento.

A palavra-chave "Protegido" (*Protected*) especifica na instrução de declaração que os elementos podem ser acessados somente de dentro da mesma classe, ou de uma classe derivada dessa classe.

A palavra-chave "Amigo" (*Friend*) especifica na instrução de declaração que os elementos podem ser acessados a partir dentro mesmo *assembly*, mas não de fora dele. O nível de acesso *Friend* pode ser usado somente no módulo ou interface.

A palavra-chave "Privada" (*Private*) especifica na instrução de declaração que os elementos podem ser acessados somente de dentro da mesma classe, ou de uma classe derivada dessa classe. O QUADRO 2.9 compara os modificadores de acesso.

Modificador de acesso	**Acesso concedido nível**	**Elementos podem ser declarados com esse nível de acesso**	**Declaração de contexto no qual se pode usar esse modificador**
Public	Irrestrito: Qualquer código que possa ver um elemento público pode acessá-lo	Interfaces,Módulos, Classes,Estruturas,Membros de estrutura,Procedimentos,Propriedades, Variáveis de membro,Constantes, Enumerações,Eventos,Declarações externas, Delegates,	Arquivo fonte Namespace Interface Module Classe Estrutura
Protected	Derivacional: Código na classe que declara um elemento protegido, ou uma classe derivada dele, pode acessar o elemento	Interfaces,Classes, Estruturas,Procedimentos, Propriedades,Variáveis de membro,Constantes,Enumerações,Eventos, Declaraçõs externas,Delegates,	Classe
Friend	Assembly: Código no conjunto de módulos (assembly) que declara que um elemento de amigo pode acessá-lo	Interfaces,Módulos,Classes,Estruturas, Membros de estrutura,Procedimentos, Propriedades,Variáveis de membro,Constantes,Enumerações, Eventos,Declarações externas,Delegates	Arquivo fonte Namespace Interface Module Classe Estrutura
Protected Friend	União de Protected e Friend: Código na mesma classe ou o mesmo conjunto como um elemento Amigo Protegido, ou em qualquer classe derivada da classe do elemento, pode acessá-lo	Interfaces, Classes, Estruturas, Procedimentos, Propriedades,Variáveis de membro, Constantes, Enumerações, Eventos, Declarações externas, Delegates	Classe
Private	Contexto da Declaração. Código de tipo que declara um elemento particular, incluindo o código de tipos contidos, pode acessar o elemento	Interfaces,Classes,Estruturas,Membros de estrutura,Procedimentos,Propriedades, Variáveis de membro,Constantes, Enumerações,Eventos,Declarações externas, Delegates,	Module Classe Estrutura

QUADRO 2.9 - Comparação entre os níveis de acesso dos elementos de programação

2.6.3. Estrutura das variáveis no *Visual Basic*

Freqüentemente surge a necessidade de se guardar valores quando executa se executa cálculos com *Visual Basic*. Por exemplo, pode ser necessário calcular vários valores, compará-los e executar diferentes operações com eles, dependendo do resultado da comparação. Os valores devem ser guardados para uma posterior comparação ou utilização em cálculos futuros.

O *Visual Basic* utiliza a variável, assim como a maioria das linguagens de programação; para guardar valores. Uma variável possui um nome (uma palavra utilizada para se referir ao valor que a variável contém) e também um tipo de dados (que determina o tipo de dado que a variável pode armazenar). Uma variável pode representar uma matriz se ela tiver que armazenar um conjunto indexado de dados relacionados.

As declarações de atribuição para executar cálculos e atribuir o resultado a uma variável são como o exemplo seguinte demonstra:

Ex.:
applesSold = 10
applesSold = applesSold + 1

O sinal de igualdade "=" do exemplo é um operador de atribuição, e não um operador de igualdade. O valor está sendo atribuído à variável "*applesSold*".

Uma variável é declarada para especificar seu nome e as características. Seu local e conteúdo determinar características da variável.
Uma variável local é aquela que está declarada dentro de um procedimento. Uma variável "de membro para membro" é declarada no nível Módulo (*Module*), dentro de uma classe, estrutura ou módulo, mas não dentro de qualquer procedimento interno para essa classe (*Class*), estrutura (*Structure*) ou módulo (*Module*).

Em uma classe ou estrutura, a categoria de uma variável de membro depende ou não que ele for compartilhado. Se for declarada como "compartilhada" (*Shared*), ela é uma variável compartilhada, e existe em uma única cópia compartilhada entre todas as instâncias da classe ou estrutura. Caso contrário, uma cópia separada da variável é criada para cada instância da classe ou estrutura. Uma cópia de uma variável da instância fornecida está disponível somente para a instância para o qual a variável foi criada. Ela é independente de uma cópia em qualquer outra instância.

O como a cláusula na instrução de declaração permite que se defina o tipo de dados ou tipo de objeto da variável , pode-se especificar qualquer um dos seguintes tipos de uma variável:

- Um tipo de dados elementar, como *Decimal*, *Boolean*, *Long* ou *Double*
- Um tipo de dados composto, como uma matriz ou estrutura
- Um tipo de objeto, ou classe, definida em seu aplicativo ou no outro aplicativo
- Uma classe *.NET Framework*, como *Labe*l ou *TextBox*
- Um tipo de interface, como *IComparable* ou *Idisposable*

O *Visual Basic* permite que diversas variáveis podem ser declaradas em uma instrução sem ter que repetir a tipo de dados.

O escopo de uma variável é o conjunto de todo o código que pode referir-se a ela sem a qualificação de seu nome (da variável). Uma O escopo é determinado pelo local aonde a variável é declarada (dentro de que estrutura, como módulo, formulário ou classe). Um código localizado em uma determinada região pode usar as variáveis definidas nessa região sem precisar qualificar seus nomes.

2.6.4. Estrutura das matrizes no *Visual Basic*

Quando matrizes são utilizadas ,o programador pode referir-se a vários valores com o mesmo nome, usando um número chamado de índice ou subscrito para diferenciá-los de um outro. Matrizes podem diminuir e simplificar o seu código, permitindo a criação de *loops* (rotinas cíclicas)que lidam com eficiência com qualquer número de elementos.

Uma Matriz (*Array*) é um conjunto de valores,que são logicamente inter-relacionados.

Uma matriz permite referir a esses valores relacionados com o mesmo nome e usar um número, chamado um índice (*index*) ou subscrito, para informá-los separados. Os valores individuais são chamados de elementos (*elements*) da matriz.

Uma Dimensão de uma matriz é uma direção na qual se pode variar a especificação de uma matriz de elementos. O número de dimensões que tem uma matriz é chamado de classificação.

Um elemento de uma matriz é especificado fornecendo um *index* ou *Subscript* para cada uma das suas dimensões. Os elementos são contíguos ao longo de cada dimensão de índice até o índice mais alto para a dimensão.

As figuras 2.7,2.8, 2.9 a seguir, mostram a estrutura conceitual de matrizes com diferentes classificações. Cada elemento nas ilustrações mostra os valores de índice que a acessam.

FIG. 2.7- Ilustração da organização de um matriz Unidimensional –Fonte: Microsoft (2009)

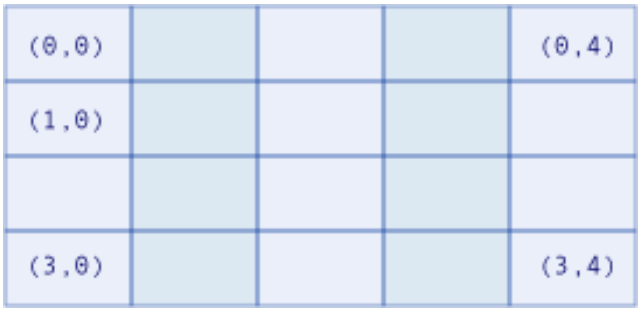


FIG. 2.8- Ilustração da organização de um matriz Bidimensional –Fonte: Microsoft (2009)

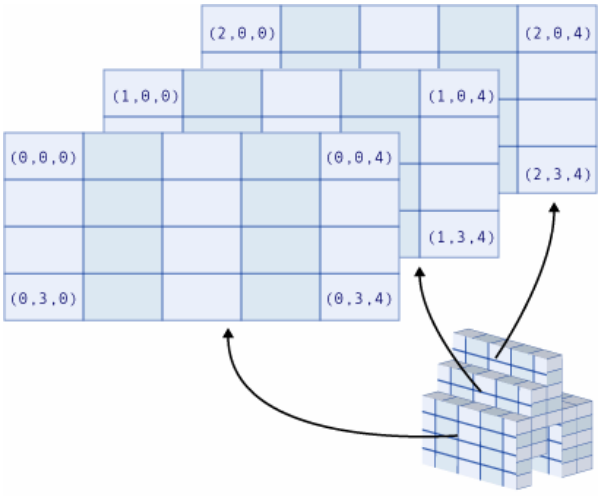


FIG. 2.9- Ilustração da organização de um matriz Tridimensional –Fonte: Microsoft (2009)

Muitas matrizes possuem apenas uma dimensão. O único requisito para especificar um elemento é a ordem para a qual o elemento contém a contagem. Portanto, esse uma matriz usa apenas um índice. Algumas matrizes possuem duas dimensões. A especificação de um elemento requer dois índices.

Existem também matrizes que possuem três dimensões, tais como valores em espaço tridimensional. Tal uma matriz usa três índices, que nesse caso representam as coordenadas *X*, *Y* e *Z* do espaço físico. Embora uma matriz possa ter até 32 dimensões, é raro se ter aplicações para mais de três dimensões.

2.6.5. Objetos e classes no *Visual Basic*: Programação orientada ao objeto

Um objeto é uma estrutura quem contém dados e métodos para manipular os dados. As palavras "classe" e "objeto" são tão usadas em programação orientada a objeto, porém é fácil confundir os termos. De modo geral, uma classe (*class*) é uma representação abstrata de algo, enquanto um objeto é um exemplo manipulável daquilo que a classe representa. A única exceção a esta regra são os membros da classe compartilhados, que são manipuláveis em ambas as instâncias de uma classe e variáveis de objeto declaradas como o tipo da classe.

Classes consistem de campos, propriedades, métodos e eventos. Campos e propriedades representam informações que um objeto contém. Campos são como variáveis, porque eles podem ser lidos ou alterados diretamente.

Propriedades são recuperadas e modificadas como campos, mas são implementadas pelos procedimentos *Get* e *Set*. Estes fornecem maior controle sobre como valores são alterados ou recuperados. A camada indireta entre o valor sendo armazenado e os procedimentos que usam este valor ajuda a isolar seus dados e permite validar valores antes de eles serem atribuídos ou recuperados.

Métodos representam ações que um objeto pode realizar. Métodos são definidos ao adicionar procedimentos, quer sejam Sub rotinas ou funções, a sua classe.

Eventos são notificações que um objeto recebe de, ou transmite para, outros objetos ou aplicativos. Eventos permitem que objetos realizem ações sempre que uma ocorrência específica ocorrer. Como o *Microsoft Windows* é um sistema operacional movido por eventos, eventos podem vir de outros objetos, aplicativos ou de entradas do usuário como cliques do mouse ou teclas pressionadas.

Campos, propriedades, métodos e eventos são apenas partes da equação da programação orientada a objeto. A verdadeira programação orientada a objeto requer objetos para dar suporte a três qualidades: encapsulamento, herança e polimorfismo.

Encapsulamento significa que um grupo de propriedades, métodos e outros membros relacionados; são tratados como unidade ou um objeto único. Objetos podem controlar como as propriedades são alteradas e os métodos são executados. Por exemplo, um objeto pode validar valores antes de permitir que as propriedades mudem.

Encapsulamento também torna mais fácil de alterar a implementação em uma ocasião mais tarde permitindo que se oculte detalhes da implementação de seus objetos, uma prática conhecida como ocultamento de dados.

Herança descreve a habilidade de se criarem novas classes baseadas em uma classe pré-existente. A nova classe herda todas as propriedades, métodos e eventos da classe base, e pode ser customizada com propriedades e métodos adicionais.

Polimorfismo significa que podem existir múltiplas classes que podem ser usadas com intercambiabilidade, mesmo que cada classe implemente as mesmas propriedades e métodos de modos diferentes. Polimorfismo é importante para a programação orientada a objeto porque permite usar itens com o mesmo nome, independente de qual tipo de objeto está em uso no momento.

2.6.6. Procedimentos no *Visual Basic*

Um procedimento é um bloco de declarações *Visual Basic* cercadas por uma declaração (*Function,Sub,Operator,Get,Set*) e uma declaração *End* correspondente. Todas as declarações executáveis em *Visual Basic* devem estar no interior de algum procedimento.

Um procedimento é invocado de algum outro lugar no código. É conhecido como uma chamada de procedimento. Quando o procedimento termina de executar, ele retorna o controle para o código que o invocou, que é conhecido como código de chamada. O código de chamada é uma declaração, ou uma expressão no interior de uma declaração, que especifica o procedimento pelo nome e o transfere o controle.

Um procedimento retorna o controle para o código de chamada quando ele termina de executar. O controle é então passado para o código de chamada seguindo o ponto de chamada do procedimento.

Com uma declaração *Return*. o controle retorna imediatamente para o código de chamada. Instruções não são executadas após a instrução *Return*.

Com uma declaração *Exit Sub* ou *Exit Function*. o controle retorna imediatamente para o código de chamada. Instruções após a instrução *Exit* não são executadas.

Se um procedimento não tem declarações *Return* ou *Exit*, conclui-se com uma declaração *End Sub* ou *End Function*, declaração *End Get* ou *End Set* seguindo a última declaração do corpo do procedimento. A declaração *End* retorna imediatamente o controle para o código de chamada. É permitida apenas uma declaração *End* em um procedimento.

Na maioria dos casos, um procedimento precisa operar em diferentes dados cada vez que ele é chamado. O programador pode passar essa informação para o procedimento como parte da chamada de procedimento. O procedimento define zero ou mais parâmetros, cada um representando um valor que se espera que seja passado. Correspondendo a cada parâmetro na definição de procedimento há um argumento na chamada de procedimento.

Um argumento representa o valor que você passa para o parâmetro correspondente em uma dada chamada de procedimento. O *Visual Basic* usa vários tipos de procedimentos:

- Subprocedimentos: Realiza ações mas não retorna um valor para o código de chamada.
- Procedimentos de tratamento de exceção: São procedimentos *Sub,* que executam em resposta a um evento lançado por ação do usuário ou por uma ocorrência em um programa.
- Procedimentos de função: Retornam um valor para o código de chamada. Eles podem realizar outras ações antes de retornar.
- Procedimentos de Propriedade retorna e atribui valores de propriedades em objetos ou módulos.
- Procedimentos de operador: Definem o comportamento de um operador padrão quando um ou mais operadores são definidos dentro de uma classe ou estrutura.
- Procedimentos Genéricos: Definem um ou mais parâmetros de tipo além de seus parâmetros normais, de tal forma que o código de chamada pode passar tipos de dado específicos cada vez que uma chamada é feita.

Cada linha de código executável em seu aplicativo deve estar no interior de algum procedimento, tal como *Main*, *calculate*,ou *Button1_Click*. Se os procedimentos forem subdivididos em procedimentos menores, sua aplicação se torna mais legível.

Procedimentos são úteis para realizar tarefas compartilhadas ou repetidas, tais como cálculos freqüentemente usados, manipulação e controle de texto, e operações de banco de dados. Um procedimento pode ser chamado de vários lugares diferentes do código, sendo assim você pode-se usar procedimentos como blocos de construção para um aplicativo.Estruturar um código com procedimentos traz os seguintes benefícios:

- Procedimentos permitem dividir seus programas em unidades lógicas discretas. Permite depurar unidades separadas mais facilmente do que depurar um programa inteiro sem procedimentos.
- Depois de desenvolver procedimentos para uso em um programa, eles podem ser usados em outros programas, freqüentemente com pouca ou nenhuma alteração.

2.6.7. Fluxo de controle no *Visual Basic*

Se deixado sem regulação, um programa executa suas instruções do início ao fim. Alguns programas muito simples podem ser escritos somente com esse fluxo unidirecional. No entanto, grande parte do poder e utilidade de qualquer linguagem de programação vem da capacidade de alterar a ordem de execução com instruções de controle e *loops*.
Estruturas de controle permitem que você regule o fluxo de execução do seu programa.

Usando estruturas de controle, pode ser escrito um código no *Visual Basic* que toma decisões ou que repete ações. Outras estruturas de controle permitem garantir disponibilidade de um recurso ou execute uma série de instruções sobre a mesma referência de objeto. As palavras-chave da linguagem *Visual Basic* e membros de biblioteca de tempo de execução são organizados por propósito e uso conforme QUADRO 2.10 abaixo.

Ação	Elemento de linguagem
Ramificar, bifurcar	***GoTo, Error***
Saia ou pause o programa.	**Finalizar, Sair, Parar**
***Loop* (rotina cíclica)**	***Do...Loop, For...Next, For Each...Next, While...End While, With***
Tome decisões.	***Choose, If...Then...Else, Select Case, Switch***
Use os procedimentos.	***Call, Function, Property, Sub***

QUADRO 2.10 - Tipos de ação de controle na programação em *Visual Basic*

2.6.8 Automação na programação: *ActiveX* e *AutoCAD VBA*

A tecnologia de automação *Active X* do *AutoCAD* permite a utilização deste programa sob a forma de um sistema servo-cliente (*serv-client system*). Neste sistema, um *software* dentre os da figura 2.10 destinado à programação (elaboração de programas, rotinas, procedimentos, operações lógicas e matemáticas, dentre outras funções), age como cliente enviando comandos para o *AutoCAD*, que por sua vez como escravo, realiza suas funções como se o usuário estivesse comandando de sua própria interface, sem a mudança de *software.* Não só este comando é permitido através da tecnologia *Active X,* mas também o compartilhamento dos dados fornecidos ou exportação dos dados para outros formatos, de sorte que sejam sempre acessíveis em termos de "linguagem" a todos os programas clientes e escravos que compõem a automação.

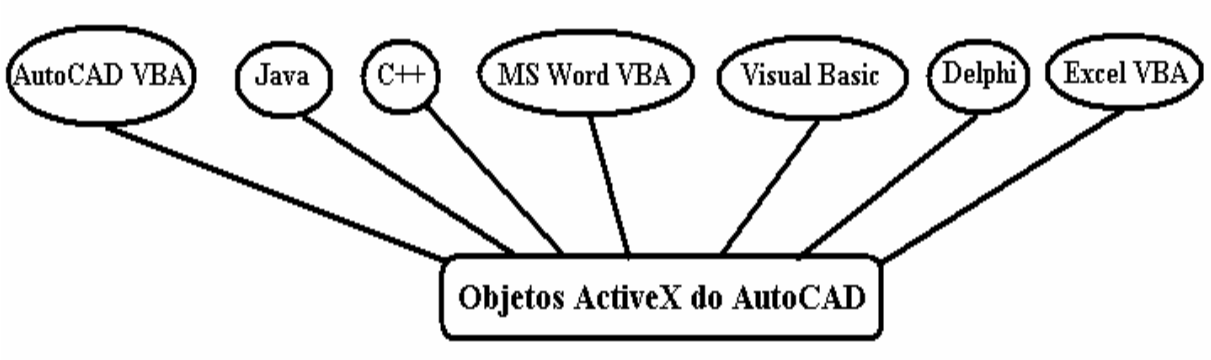


FIG. 2.10- ***Softwares clientes com interface para o AutoCAD***.

A princípio a prática de se mudar de *software* não é vantajosa, uma vez que as mesmas operações realizadas são as mesmas. Quando se considera a utilização de um software que tem uma interface ao usuário e que este realiza uma seqüência consideravelmente longa, muitas vezes repetitivas de comandos, deseja-se que o *software* realize a série de operações automaticamente, ou seja, consiste em uma automação desse processo.

Na maior parte dos casos nos quais a automação é desejável, mesmo que a seqüência de operações seja a mesma ou similar, muitos ou todos os parâmetros mudam; e é necessário que o *software* busque os dados e tenha meios de utilizá-los quando for pertinente fazê-lo. Desse modo espera-se reduzir ao máximo a necessidade da intervenção do usuário e assim se espera que o resultado seja atingido em tempo reduzido e com as falhas minimizadas ou eliminadas.

O desenvolvimento do *software* proposto neste trabalho implica em considerar as diversas nuances citada, pois é de se esperar que este funcione para gerar o código numérico para peças de geometrias variadas e dotadas de certa complexidade até; de permitir o usuário definir como será feito o corte (quantos cortes, em que plano; etc...); portanto a automação dos procedimentos se depara com esse tipo de problema de concepção, ou seja, do que se deseja do *software.*

Outra questão a ser levantada a esse respeito é a capacidade do programa, a plataforma de desenvolvimento; de efetivamente realizar através das suas possibilidades o que é objetivado. Nesse ponto é possível que se torne inviável realizar o propósito quando se esgotam seus recursos para realizar determinada tarefa e, por conseguinte se esbarra nas limitações do *software*.

2.6.8.1. *AutoCAD VBA*

Dentro da tecnologia de automação *Active X,* encontra-se inserido o *AutoCAD VBA*, que nada mais é do que um ambiente de programação orientado ao objeto, destinado a proporcionar uma capacidade de desenvolvimento de *software* semelhante à do *Visual Basic*. De fato o que ocorre é que esse ambiente promove e transcrição das linguagens, propiciando a utilização de programação similar à que é feita no *Visual Basic*. Por conseguinte os comandos do *AutoCAD* podem ser realizados segundo a sintaxe do *Visual Basic* e são "entendidos" segundo a sintaxe do *AutoCAD*. Por conseguinte o usuário pode realizar a automação de tarefas no *AutoCAD*, manipulando-o externamente como se o fizesse diretamente no próprio *software.* É importante citar que nem todas as funcionalidades (também conhecidas como ferramentas) disponíveis na interface do *AutoCAD* estão disponíveis na tecnologia *VBA* e portanto automatizáveis.

2.6.8.2. Tecnologia *Active X*

A tecnologia *Active X* é por definição uma programação orientada ao objeto, portanto ela tem a programação orientada ao objeto como raiz para todo o tipo de automação realizada. Existem diversos tipos objetos dentro deste meio. Temos os objetos gráficos como linhas, arcos, textos e cotas (dimensões).

Também as estruturas organizacionais como as *layers* (linhas com as quais os objetos são representados)*, groups* (grupos) e *blocks* (blocos: conjuntos de objetos gráficos); são objetos. Existem os padrões de estilos para os tipos de *layers e* cotas*,* por exemplo*,* e dando continuidade podem ser citados os modos de visualização dos desenhos, o arquivo do desenho e o próprio *AutoCAD* para aplicação; todos considerados objetos dentro da tecnologia *Active X.*

A figura do Anexo I, relaciona os diversos objetos que constituem a tecnologia de automação em questão. Os objetos podem ser divididos em categorias. Os objetos de aplicação (*Application Objects*), organizados conforme FIG. 2.11, são os que; a partir deles é permitido ter o acesso a qualquer outro objeto, método ou propriedade. Os objetos do tipo "documento", consistem em um desenho, um arquivo de desenho do *AutoCAD*, e é encontrado na coleção (collection) de documentos. Esses objetos permitem o acesso maioria dos objetos gráficos e não-gráficos. São eles os objetos gráficos, objetos não-gráficos e as preferências.

Os objetos gráficos também denominados entidades (*entities*) dentro deste contexto são os objetos visíveis. A eles são aplicáveis métodos como copiar, colar, apagar e espelhar. Alguns objetos desses possuem propriedades específicas como área, perímetro e raio.

Os objetos não-gráficos são meramente informativos, são estruturas organizacionais conforme foi citado anteriormente. Por exemplo podem ser mencionados os tipos de linhas (*Layers*) e seleções de objetos (*Selection Sets*). Estes também têm seus métodos e propriedades específicas.

Em seguida serão tratados os objetos de preferência. Estes objetos de preferência estão relacionados às opções que o programa disponibiliza ao usuário adequar a interface do programa a seu modo; os objetos de impressão que controlam as preferências de impressão dos desenhos e os de utilidade, que por sua vez controlam a interface, porém focada na a entrada de dados por parte do usuário por intermédio da linha de comando do próprio *AutoCAD.*

2.6.8.3 Enfoque nos objetos gráficos da tecnologia *Active X*

Dentro da aplicação proposta o interesse especial é voltado aos objetos gráficos. Deles interessam dados que podem ser extraídos ou mesmo obtidos através dos objetos bi e tridimensionais passíveis de serem desenvolvidos com o *software.* Os objetos de maior interesse são: as retas, círculos, arcos, *splines* e regiões, como elementos bidimensionais; e os sólidos, como elementos tridimensionais.

As retas são os objetos mais básicos que constituem uma classe denominada *AcadLine* dentro do ambiente de programação. Uma classe nada mais é que um conjunto de objetos com propriedades em comum. Podem ser criados diferentes tipos de retas, semi-retas, segmentos de retas e linhas múltiplas que podem ser associadas a arcos formando um objeto do tipo *Polyline.*

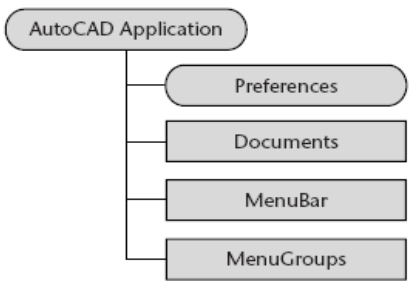


FIG. 2.11- Disposição das categorias de objetos de aplicação

Assim como os demais objetos gráficos, as retas podem ser inseridas em diferentes objetos de hierarquia superior e por tanto mais abrangentes como os blocos (*blocks*). Para criar, acessar ou modificar uma reta (o que implica em lidar com seus dados), é necessário fazê-lo utilizando os objetos do QUADRO 2.11.

Nome da classe VBA	**AcadLine**
Objeto é criado usando:	*ModelSpace.AddLin* *PaperSpace.AddLine* *Block.AddLine*
Objeto é acessado usando:	*ModelSpace.ItemPaperSpace.Item* *Block.Item* *SelectionSet.Item* *Group.Item*

QUADRO 2.11 - Objetos que permitem o acesso a propriedades, métodos e eventos das retas

Estão associados Às retas propriedades (properties), métodos (method) e eventos (*event*); alguns lhes são exclusivos, e no QUADRO 2.12, abaixo, todos eles estão relacionados.

Em se tratando das propriedades das retas, pode-se afirmar o interesse especial em duas delas para a aplicação do *software* desenvolvido; o ponto inicial (*StartPoint*) e final (*EndPoint*) da reta, que são responsáveis pela determinação precisa da reta no plano cartesiano e por conseqüência também são responsáveis pela trajetória para a máquina de comando numérico. Uma vez conhecidos os pontos de início e fim e sabendo que se trata de uma interpolação linear, todas as informações necessárias para gerar o código numérico ficam completamente definidas.

Os círculos constituem um dos tipos de curvas passíveis de serem realizadas pelo *AutoCAD*. Basicamente, para se fazer um círculo são necessários um ponto para a localização do centro e o raio, ou seu diâmetro. Os objetos tipo círculo (da classe *AcadCircle*), podem ser criados, manipulados ou terem seus dados extraídos por meio dos mesmos objetos de hierarquia superior do que as retas (QUADRO 2.13), conforme QUADRO 2.14, porém este objeto possui propriedades e métodos diferentes dos quais as retas apresentam.

Estas propriedades e métodos não são extensivos às retas. Como exemplo temos propriedades como diâmetro e área como pode ser visto no QUADRO 2.14.

Os arcos por sua vez, podem ser resumidos a um caso particular de circunferências,pois são trechos delas, e, portanto têm propriedades além dos círculos.

Métodos	Properiedades	Eventos
ArrayRectangular *Copy* *Delete* *GetBoundingBox* *GetExtensionDictionary* *GetXData* *Highlight* *IntersectWith* *Mirror* *Mirror3D* *Move* *Offset* *Rotate* *Rotate3D* *ScaleEntity* *SetXData* *TransformBy* *Update*	*Angle* *Application* *Document* *Delta* *EndPoint* *Handle* *HasExtensionDictionary* *Hyperlinks* *Layer* *Length* *Linetype* *LinetypeScale* *Lineweight* *Normal* *ObjectID* *OwnerID* *PlotStyleName* *StartPoint* *Thickness* *TrueColor* *Visible*	*Modified*

QUADRO 2.12 - Métodos, propriedades e eventos associados a retas

Nome da classe VBA	**AcadCircle**
Objeto é criado usando:	*ModelSpace.AddCircle* *PaperSpace.AddCircle* *Block.AddCircle*
Objeto é acessado usando:	*ModelSpace.Item* *PaperSpace.Item* *Block.Item* *SelectionSet.Item* *Group.Item*

QUADRO 2.13 - Objetos que permitem o acesso a propriedades, métodos e eventos dos círculos

Imediatamente consegue-se abstrair duas dessas propriedades; o ponto de início e o do fim do arco, além do centro e raio que já são propriedades também inerentes aos círculos. Mas, nem por isso os arcos deixam de constituir uma classe.Existe uma classe para os objetos tipo arco, denominada *AcadArc* cujos meios de acesso, edição e extração de dados podem ser visualizados no QUADRO 2.15; e; as propriedades, métodos e eventos associados aos arcos podem ser contemplados no QUADRO 2.16.

Métodos	Properiedades	Eventos
ArrayPolar *ArrayRectangular* *Copy* *Delete* *GetBoundingBox* *GetExtensionDictionary* *GetXData* *Highlight* *IntersectWith* *Mirror* *Mirror3D* *Move* *Offset* *Rotate* *Rotate3D* *ScaleEntity* *SetXData* *TransformBy* *Update*	*Application* *Area* *Center* *Circumference* *Diameter* *Document* *Handle* *HasExtensionDictionary* *Hyperlinks* *Layer* *Linetype* *LinetypeScale* *Lineweight* *Normal* *ObjectID* *OwnerID* *PlotStyleName* *Radius* *Thickness* *TrueColor* *Visible*	*Modified*

QUADRO 2.14 - Objetos que permitem o acesso a propriedades, métodos e eventos dos círculos

Nome da classe VBA	AcadArc
Objeto é criado usando:	*ModelSpace.AddArc* *PaperSpace.AddArc* *Block.AddArc*
Objeto é acessado usando:	*ModelSpace.Item* *PaperSpace.Item* *Block.Item* *SelectionSet.Item* *Group.Item*

QUADRO 2.15 - Objetos que permitem o acesso a propriedades, métodos e eventos dos círculos

Os objetos gráficos chamados *Splines*, da classe *AcadSpline*, são curvas ajustadas entre pontos pré-estabelecidos, conhecidos como pontos de controle (*ControlPoints*) por curvas de ajuste de caráter não uniforme (*NURBS*). Neste ajuste, essa classe permite a determinação da tolerância do ajuste feito entre os pontos de controle, ou seja, o quão próxima à curva está desses pontos. A seguir, como foi feito para os outros objetos, serão citados tanto

os meios de acesso, edição e extração de dados (QUADRO 2.17), quanto suas propriedades, métodos e eventos (QUADRO 2.18).

Métodos	Properiedades	Eventos
ArrayPolar *ArrayRectangular* *Copy* *Delete* *GetBoundingBox* *GetExtensionDictionary* *GetXData* *Highlight* *IntersectWith* *Mirror* *Mirror3D* *Move* *Offset* *Rotate* *Rotate3D* *ScaleEntity* *SetXData* *TransformBy* *Update*	*Application* *ArcLength* *Area* *Center* *Document* *EndAngle* *EndPoint* *Handle* *HasExtensionDictionary* *Hyperlinks* *Layer* *Linetype* *LinetypeScale* *Lineweight* *Normal* *ObjectID* *ObjectName* *OwnerID* *PlotStyleName* *Radius* *StartAngle* *StartPoint* *Thickness* *TotalAngle* *TrueColor* *Visible*	*Modified*

QUADRO 2.16 - Objetos que permitem o acesso a propriedades, métodos e eventos dos arcos

Nome da classe VBA	AcadSpline
Objeto é criado usando:	*ModelSpace.AddSpline* *PaperSpace.AddSpline* *Block.AddSpline*
Objeto é acessado usando:	*ModelSpace.Item* *PaperSpace.Item* *Block.Item* *SelectionSet.Item* *Group.Item*

QUADRO 2.17 - Objetos que permitem o acesso a propriedades, métodos e eventos das *Splines.*

Métodos	Properiedades	Eventos
AddFitPoint	Application	*Modified*
ArrayPolar	Area	
ArrayRectangular	Closed	
Copy	ControlPoints	
DeleteFitPoint	Degree	
ElevateOrder	Document	
Delete	EndTangent	
GetBoundingBox	FitPoints	
GetControlPoint	FitTolerance	
GetExtensionDictionary	Handle	
GetFitPoint	HasExtensionDictionary	
GetWeight	Hyperlinks	
GetXData	IsPeriodic	
Highlight	IsPlanar	
IntersectWith	IsRational	
Mirror	Knots	
Mirror3D	Layer	
Move	Linetype	
Offset	LinetypeScale	
PurgeFitData	Lineweight	
Reverse	NumberOfControlPoints	
Rotate	NumberOfFitPoints	
Rotate3D	ObjectID	
ScaleEntity	OwnerID	
SetControlPoint	StartTangent	
SetFitPoint	TrueColor	
SetWeight	Visible	
SetXData	Weights	
TransformBy		
Update		

QUADRO 2.18 - Objetos que permitem o acesso a propriedades, métodos e eventos das *Splines.*

Em uma hierarquia superior dos objetos gráficos bidimensionais citados, desde que estes formem um contorno fechado, podem vir a constituir um objeto do tipo região, da classe *AcadRegion*. Essa classe de objetos tem algumas propriedades bastante úteis como área, pontos limite mínimo e máximo (no sistema de coordenadas global, absoluto, do *software*) da região, momento de inércia e o centróide. Essas e outras propriedades são aplicáveis, por exemplo, na diferenciação dos contornos. Cada contorno fechado mesmo no mesmo plano é uma região, desta maneira uma região pode conter mais de um contorno fechado dentro de si. Dentro dessa lógica podem os diferentes contornos inerentes à região, serem identificados por sua área e/ ou centróide ou alguma outra propriedade desse objeto que seja conveniente. Tais propriedades (QUADRO 2.19) podem ser acessadas através dos objetos relacionados no QUADRO 2.20.As regiões também podem ser obtidas através da interseção entre sólidos e planos através de métodos próprios para objetos da classe *AcadSolid,* que englobam objetos gráficos tridimensionais .

Métodos	Properiedades	Eventos
ArrayPolar *ArrayRectangular* *Boolean* *Copy* *Delete* *Explode* *GetBoundingBox* *GetExtensionDictionary* *GetXData* *Highlight* *IntersectWith* *Mirror* *Mirror3D* *Move* *Rotate* *Rotate3D* *ScaleEntity* *SetXData* *TransformBy* *Update*	*Application* *Area* *Centroid* *Document* *Handle* *HasExtensionDictionary* *Hyperlinks* *Layer* *Linetype* *LinetypeScale* *Lineweight* *MomentOfInertia* *Normal* *ObjectID* *ObjectName* *OwnerID* *Perimeter* *PlotStyleName* *PrincipalDirections* *PrincipalMoments* *ProductOfInertia* *RadiiOfGyration* *TrueColor* *Visible*	*Modified*

QUADRO 2.19 - Objetos que permitem o acesso a propriedades, métodos e eventos das *Splines.*

Nome da classe VBA	AcadRegion
Objeto é criado usando:	*ModelSpace.AddRegion* *PaperSpace.AddRegion* *Block.AddRegion*
Objeto é acessado usando:	*ModelSpace.Item* *PaperSpace.Item* *Block.Item* *SelectionSet.Item* *Group.Item*

QUADRO 2.20 - Objetos que permitem o acesso a propriedades, métodos e eventos das regiões.

Existem outros elementos bidimensionais que possuirão a maioria das propriedades, métodos e eventos similares aos dos objetos apresentados, porém, sem sombra de dúvida; os objetos citados são necessários e suficientes para o propósito de obter os códigos para as máquinas de comando numérico.

Com relação aos sólidos, existem alguns tipos básicos desse tipo de objeto, o *AcadSolid*, e são eles: a caixa (um paralelepípedo) ,a esfera, o toro, o cone, o cone elíptico (cuja base é uma elipse), o cilindro elíptico (de modo análogo ao cone elíptico), o prisma. Através da soma ou subtração destes sólidos (da subtração de volumes comuns e adição de

volumes distintos) pode-se obter uma grande sorte de outros objetos sólidos cujas geometrias sejam bem mais complexas do que as anteriores.

Os sólidos virtuais também podem ser criados no ambiente do *AutoCAD* através da extrusão de uma região fechada (aqui não se faz necessário o conceito anterior do objeto tipo "região", mas nesse ponto considera-se um conjunto de elementos gráficos bidimensionais que formam um contorno fechado) ao longo de um vetor ou mesmo uma curva aberta que serve de guia ou caminho para a extrusão; ou ainda da rotação de tal região ou regiões no entorno de um eixo estabelecido no sistema de coordenadas.

Os métodos aplicáveis à criação dos objetos sólidos básicos são enumerados a seguir: *AddBox, AddCone , AddCylinder, AddEllipticalCone, AddEllipticalCylinder, AddExtrudedSolid, AddExtrudedSolidAlongPath, AddRevolvedSolid,AddSolid, AddSphere, AddTorus, AddWedge.*

2.6.8.4. Métodos aplicáveis aos objetos gráficos

Os métodos são ações permitidas a um determinado objeto que implicam na sua edição, modificação. Eles existem tanto para objetos gráficos quanto para os que não o são. Dentre os métodos apresentados nas figuras já expostas durante a resenha da tecnologia *ActiveX* , existem alguns que são de maior interesse para a finalidade proposta e portanto serão os que serão mais detalhados a seguir.

Aos sólidos é permitida a criação de uma seção transversal, como pode ser observado na FIG 2.12. A seção obtida através da interseção destes com o plano no qual se deseja essa seção. Ao final desse processo é criada uma região que contém objetos bidimensionais que representam as interseções entre os limites do sólido e o plano. Essa região enquadra-se no conceito do objeto tipo "região" do *AutoCAD*, e , portanto, é interpretada como as entidades classificadas como *AcadRegion.*

O método de copiar, *Copy*, permite que um ou mais objetos com as mesmas características dos selecionados sejam inseridos no mesmo local ou em local ou mesmo desenho diferente.

Semelhante ao método de copiar, existe o método de espelhar, o *Mirror*, que cria objetos gráficos bidimensionais simétricos aos selecionados em relação a uma linha de simetria definida por dois pontos arbitrados pelo usuário.

Dentro dessa mesma linha de ações, encontra-se o método de criar arranjos, o Array, que realiza basicamente o mesmo que o método de copiar, contudo o faz em uma

distribuição cartesiana ou polar, de modo a gerar um arranjo de objetos gráficos eqüidistantes uns dos outros.

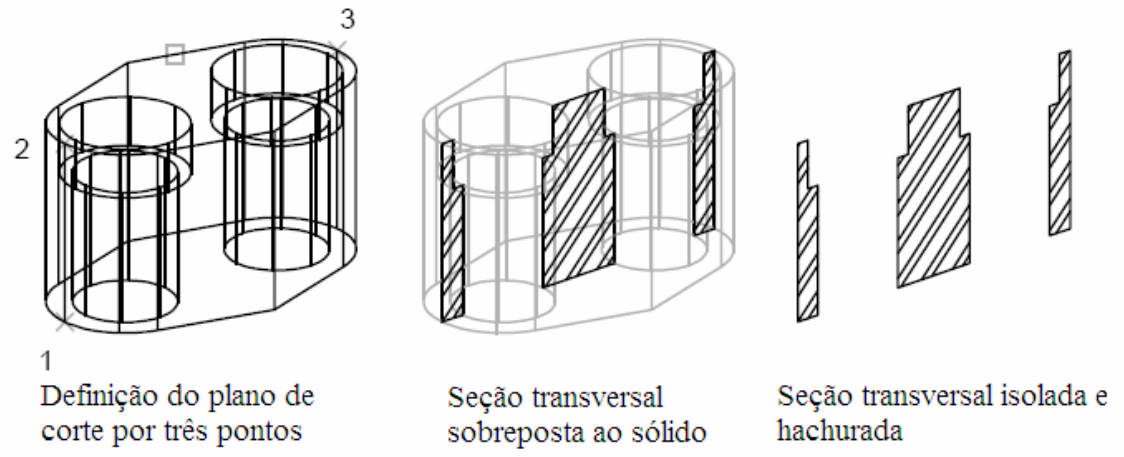


FIG. 2.12 - Exemplo de criação de uma seção transversal

Os métodos de movimentar (*Move*) e girar (*Rotate*) realizam transformações de coordenadas que modificam as posições dos objetos gráficos selecionados conforme a necessidade do usuário.

Quando se quer excluir um objeto gráfico se usa o método *Delete*, de sorte que este deixa de ser acessível (não pode ser visualizados ou alterados suas propriedades e a eles não se aplicam nenhum dos métodos).

Alguns dos objetos gráficos não permitem acesso a determinadas propriedades específicas de outros objetos igualmente gráficos que constituem estes primeiros, ou algum método também específico pode não surtir efeito. A fragmentação dos objetos gráficos em outros de menor hierarquia é realizada através do método *Explode*, que acarreta no fim tanto do objeto gráfico de hierarquia maior quanto do acesso a suas propriedades e métodos específicos.

2.6.8.5 Enfoque nos objetos não-gráficos da tecnologia *Active X*

Os objetos não gráficos são estruturas organizacionais conforme já foi tratado anteriormente. A aplicação de interesse destes objetos para o *software* desenvolvido, é capacidade de alterar propriedades dos objetos gráficos .Dentre os objetos não gráficos, existem dois que são de particular interesse dentro da aplicação proposta, as seleções e as camadas, denominadas *layers*. Devido à importâncias desses dois objetos não gráficos, estes

serão tratados em tópicos exclusivos para o melhor entendimento de como e quais propriedades são alteradas manipulando estes objetos através da tecnoogia *ActiveX*.

As seleções são agrupamentos de objetos gráficos que podem ser tratados como um único objeto. Elas são utilizadas para agrupar um conjunto de objetos aos quais se deseja aplicar um método como copiar, mover, espelhar explodir ou qualquer outro método; a todos os objetos selecionados simultaneamente.A definição das seleções se dá em duas etapas distintas. A primeira consiste na no estabelecimento (ou declaração) da seleção e em um segundo momento na adição de objetos gráficos a essa seleção.

Para discriminar quais objetos são de interesse é necessário o estabelecimento de critérios. Essa segregação será realizada por meio da adição de filtro à seleção. Os filtros são propriedades comuns aos objetos que se deseja selecionar. Para esclarecer a situação podem ser dados alguns exemplos. Pode-se fazer um apanhado de todos os círculos cujos raios sejam menores que 5 mm dentro de um desenho, de todos os objetos que tenham a *layer* denomina da *layer 1*, de todos objetos que estejam dentro de uma determinada área delimitada ou mesmo de todos objetos que estejam no desenho. Grande parte dos filtros tem códigos específicos, os códigos *DXF*. O código para o filtro de cores é 62 e a cada cor que se escolha para filtrar é atribuído um número inteiro de 0 a 256 (as cores representáveis pelo *AutoCAD*).

Quando se deseja filtrar objetos pelo seu tipo, por exemplo, na seleção deseja-se adicionar uma linha, arco, um objeto qualquer em que suas coordenadas façam parte do filtro, ou mesmo textos; então operações matemáticas e lógicas fatalmente serão aplicadas. O QUADRO 2.21 abaixo, relaciona alguns dos operadores matemáticos disponíveis. A seguir são apresentadas no QUADRO 2.22 as operações lógicas possíveis para os filtros .

Símbolo	Significado
“=”	Igual a…
“!=” “/=” “<>”	Não é igual a…
“<”	Menor que…
“<=”	Menor ou igual a…
“>”	Maior que...

QUADRO 2.21 - Exemplos de operações matemáticos permissíveis

Símbolo	Significado
“<AND”	E
“<OR”	OU
“<XOR”	OU EXCLUSIVO
“<NOT”	NÃO

QUADRO 2.22 - Exemplos de operações lógicas permissíveis

A classe das *layers* contempla as propriedades gráficas tanto visuais como de impressão. Basicamente esses objetos controlam o estilo das linhas que constituem os objetos gráficos (contínua, tracejada, etc...) bem como sua espessura e cor (FIG 2.13).

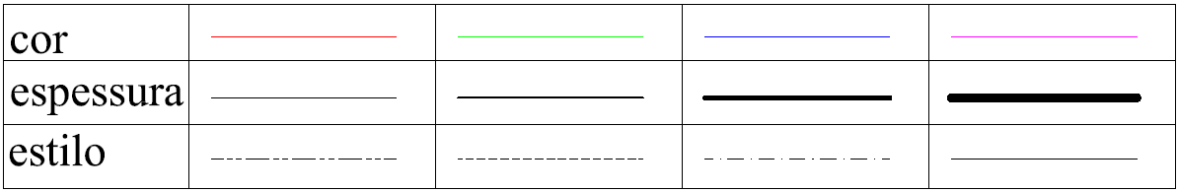


FIG. 2.13 - Exemplos de propriedades das *layers*

Mas nem todos os objetos gráficos possuem essas três propriedades, como é o caso dos objetos do tipo texto. Estes não possuem a propriedade de estilo.

Existem outras propriedades das *layers* que são de grande utilidade. Quando se faz necessário que o objeto gráfico não apareça, é utilizada a propriedade *Turn off*. Os objetos continuam existindo, mas não são visualizados e, portanto não podem selecionados na interface do visualizador. A cada modificação a visualização dos objetos cuja *layer* seja “desligada”, será atualizada. No caso do “congelamento” a atualização da *layer* a cada modificação não ocorre, é um recurso para tornar mais ágil o programa baixando seu processamento gráfico. Existe também a possibilidade de uma *layer* ser “trancada”. Para fins práticos isso significa que esta pode ser visualizada, embora qualquer objeto gráfico que tenha essa determinada *layer* não possa ser alvo de nenhum método, ou seja, este não pode ser editado.

3. METODOLOGIA

3.1. Escolha do programa CAD

Existem várias plataformas *CAD* e sistemas modeladores de sólidos comerciais com possibilidade de controle através da interface gráfica, tais como *MicroStation®, SolidWorks®, Unigraphics®, Inventor®*, etc, além do próprio *AutoCAD®*. Entretanto a escolha deste último se deveu a três fatores.

Primeiramente, o presente trabalho representa em parte a continuação e aprimoramento de uma pesquisa anterior realizada no Laboratório de Projetos Mecânicos do Departamento de Engenharia Mecânica, inserido no Programa de Pós-Graduação em Engenharia Mecânica da UFMG.

SANTOS (2002) desenvolveu um programa baseado também nas tecnologias *ActiveX* e *VBA* denominado *3DFORM 2.0*. Este programa norteia o rumo do desenvolvimento do trabalho atual, no entendimento das tecnologias *ActiveX* e *VBA*. Através da compilação do programa *3DFORM 2.0* foi possível identificar suas limitações para então elaborar estratégias para minimizar tais limitações. No atual estudo procurou-se evitar o que aconteceu com trabalho anterior que poderá ser abandonado devido à obsolescência do sistema operacional no qual o programa foi compilado, o *Windows 98®,* cada vez menos utilizado devido à evolução constante dos sistemas operacionais oferecidos.

O segundo fator levado em consideração foi que existe, ainda que reduzida, uma literatura técnica sólida acerca dessa tecnologia de automação para o *AutoCAD®* em suas várias versões. Portanto, baseando-se nessa literatura (AUTODESK,1999; AUTODESK,2003; AUTODESK,2006; FINKELSTEIN, 2007; FINKELSTEIN, 2004), pode-se encurtar o tempo necessário para o domínio da tecnologia e, por conseguinte reduzir o tempo para a sua aplicação e obtenção de alguns resultados.

O outro fator considerado foi o grande número de usuários do *AutoCAD®* em comparação com outras plataformas *CAD* e modeladores de sólidos, além do tempo em que este programa está no mercado. O fato de que há muitos grupos de discussão, fóruns, páginas na *Internet*, etc, com a difusão de conhecimento em termos de programas executáveis, código fonte, explicações sobre o funcionamento de comandos, soluções de casos e diversos aplicativos desenvolvidos; auxiliou no desenvolvimento desta pesquisa com o *AutoCAD®*.

Nesta mesma linha, pode-se citar a existência de outro argumento decorrente da consolidação deste programa *CAD* no mercado. Vários outros *softwares CAD* mais recentes foram obrigados a manter uma conectividade por assim dizer com as diversas versões do *AutoCAD*® permitindo tanto o acesso às informações dos modelos criados em outras plataformas, quanto à transcrição dos dados para seu formato de arquivo próprio, salvando e abrindo arquivos com extensão *.dwg* e *.dxf*. Assim, é possível modelar um objeto em um *software* mais apropriado ou que se tenha familiaridade e então transferir os dados para os formatos de arquivo do *AutoCAD*® (AUTODESK,2006).

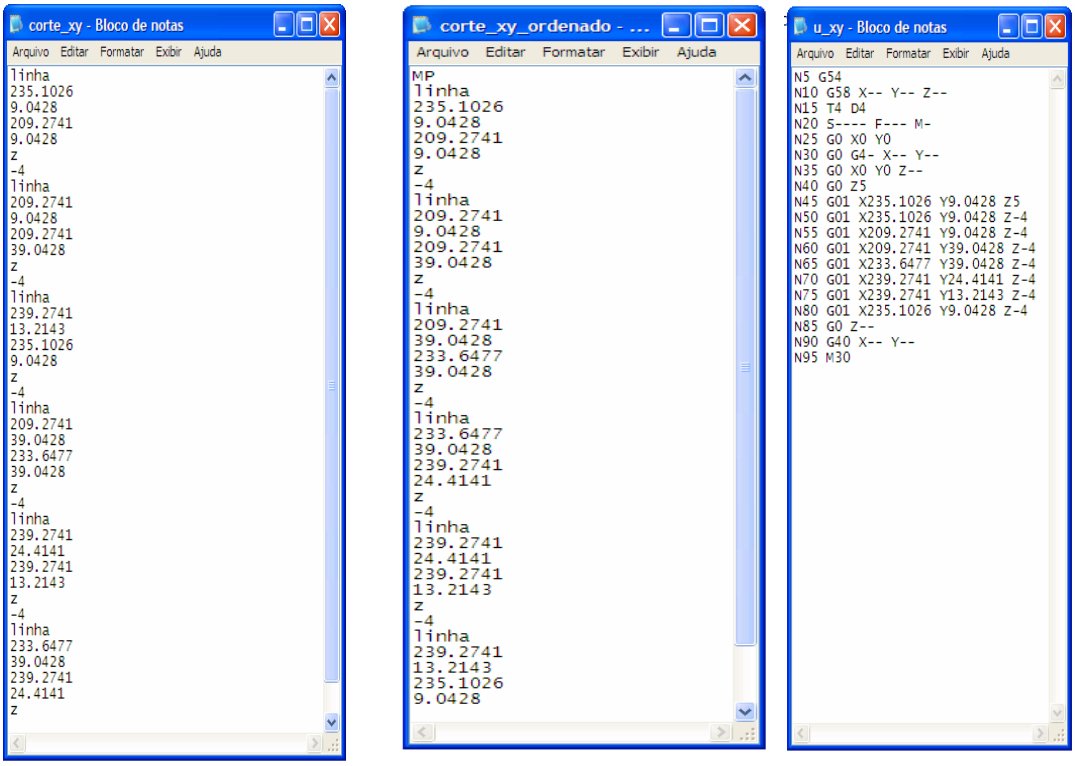
3.2. Análise do programa computacional *3DFORM 2.0*

O programa *3DFORM 2.0* teve seu código fonte analisado e foi testada sua versão executável em funcionamento para o entendimento de sua lógica, identificação de limitações e oportunidades de aprimoramento. Foi identificado que o programa possui duas funcionalidades: a modelagem de protótipos virtuais através de operações booleanas com sólidos primitivos como cones, esferas e cilindros, e a geração de códigos de comando numérico para usinagem a partir de dados retirados do protótipo virtual; ambas realizadas através do controle externo do AutoCAD. Como o foco atual é a geração de código, a análise sobre modelagem no programa não será contemplada.

Com relação à geração do código de comando numérico, foi observado que as coordenadas extraídas do modelo 3D eram armazenadas em variáveis de tamanho fixo e então gravadas em arquivos de texto. Os processos de extração de dados do modelo virtual, sua ordenação e geração do código segundo a sintaxe convencional a usinagem eram realizados utilizando um total de três arquivos. O primeiro arquivo indicado na FIG. 3.1(a) armazena os nomes atribuídos aos objetos gráficos bidimensionais como linhas, arcos e círculos, seguidos de suas coordenadas extraídas. O processo de ordenação ocorre baseado neste primeiro arquivo. Os dados são armazenados em uma matriz de tamanho fixo, capaz de armazenar cinco mil (5000) objetos gráficos bidimensionais considerados pelo programa, são eles: linhas, arcos, círculos e *splines*. Dentro dessa matriz a lógica de ordenação é executada gerando por sua vez outro arquivo de texto como o ilustrado na figura 3.1(b). Uma vez ordenados os dados o *software* então gera o código de comando numérico armazenando-o em um terceiro arquivo como ilustrado na figura. 3.1(c).

A operação de fatiamento do modelo 3D, que consiste na interseção do sólido por planos, e conseqüente geração de objetos bidimensionais; é permitida para os planos *XY*, *YZ* e

ZX. A interface do programa permite também que o usuário defina as coordenadas espaciais de início e fim do fatiamento bem como número desejado de camadas, figura 3.2.



(a) **(b)** **(c)**

FIG. 3.1-(a) Arquivo gerado pela compilação do *software 3DFORM 2.0* contendo as coordenadas extraídas dos objetos gerados pelo fatiamento do modelo 3D. (b) Arquivo gerado pela compilação do *software 3DFORM 2.0* contendo as mesmas coordenadas já ordenadas. (c) Código numérico referente às coordenadas presentes nos arquivos das figuras 3.1(a) e (b).

FIG. 3.2- Tela da interface com os parâmetros de entrada para o usuário do *software 3DFORM 2.0*

Da análise realizada no programa *3DFORM 2.0* foi possível identificar as seguintes oportunidades de melhorias para o desenvolvimento de um novo programa:

- São utilizados três arquivos de texto até chegar ao código numérico que poderiam ser substituídos por apenas um contendo apenas o código numérico.
- A matriz utilizada no processo de ordenação do *3DFORM 2.0* é definida estaticamente, o que limita a quantidade de objetos passíveis de serem armazenados. Não se sabe a priori a quantidade de linhas ou círculos, por exemplo, presentes em uma mesma camada (confinados no mesmo plano) ou mesmo em todas as camadas. Desta maneira pode ocorrer falha na execução por insuficiência de memória para armazenar uma quantidade superior a dimensionada. Em contrapartida, para modelos simples, a matriz fica superdimensionada, o que é um aumento desnecessário no custo computacional e assim prejudica o desempenho do programa. Para resolver esta limitação é proposta a utilização de uma matriz definida por alocação dinâmica de memória, que reservaria apenas o espaço necessário para armazenar os dados.
- Ainda com relação ao processo de ordenação dos objetos bidimensionais sugere-se sua realização a cada camada e não em todas as camadas simultaneamente como é feito. O *3DFORM 2.0,* após o processo de fatiamento em camadas do modelo virtual gera, por exemplo, retas e arcos a cada camada e realiza a ordenação de todas ao mesmo tempo. Propõe-se a realização da ordenação dos objetos gerados a cada fatiamento do modelo 3D. Adotando esta prática, um nível de ordenação é eliminado, pois como todos os objetos estão no mesmo plano, ou seja, estão na mesma camada e para todos os efeitos possuem uma das três coordenadas em comum e que portanto não necessita de ser ordenada entre eles.

Quanto aos parâmetros do usuário é válido ressaltar que a estratégia utilizada pelo *3DFORM 2.0* obriga o usuário a conhecer não só as dimensões do objeto, mas também a posição do mesmo em termos de coordenadas no espaço virtual. No novo programa planeja-se fornecer meios ao usuário de posicionar de maneira prática o modelo em relação em uma posição desejada em relação à origem do sistema de coordenadas do espaço virtual. A importância desse aspecto se revela na estratégia já adotada no *3DFORM 2.0* de se trabalhar

com coordenadas absolutas no código numérico e que pretende ser mantida. Neste tipo de abordagem todas as coordenadas são influenciadas pelo posicionamento do modelo em relação à origem considerada para o sistema de coordenadas (ASM, 1999).

3.3. O programa computacional *AutoCAM*

A partir da análise do *3DFORM 2.0* e formadas algumas condições de contorno para a elaboração de um novo programa, foram realizados vários estudos no sentido de empregar corretamente a sintaxe necessária à utilização dos recursos disponíveis na execução de procedimentos atribuídos ao programa em desenvolvimento. São exemplos desses procedimentos: viabilidade técnica de se implementar as ações propostas, utilizando as ferramentas disponíveis, tanto na linguagem de programação quanto no ambiente de programação.(*VB* e *AutoCAD VBA*); o controle do modo de visualização do modelo, a lógica por trás da interface com o usuário,e a gravação e visualização do arquivo de texto.

O primeiro item pesquisado foi o método de fatiamento do modelo virtual 3D em camadas. Este método gera uma série de objetos bidimensionais como retas, arcos e círculos, como interseção de planos com os limites do sólido no plano. A estratégia elaborada visou fatiar o modelo CAD em camadas de mesma espessura ao longo do eixo ortogonal ao plano de fatiamento escolhido abrangendo todo o modelo. A prática de se fatiar o modelo em camadas iguais é semelhante à estratégia empregada nas tecnologias de prototipagem rápida de modo geral, abordada por PARK,TARI, HAHN (2000); MORGAN,SUTCLIFFE, O'NEILL (2001) YANG (2002), BELLINI, GÜÇERI (2003) e TONG.,LEHTIHET,JOSHI (2003). Para viabilizar a execução do fatiamento de um modelo *CAD*, enxergou-se a necessidade de conhecer as dimensões e definir a localização do modelo no espaço virtual; e da existência da ferramenta (comando) do *AutoCAD*® que realize o fatiamento e da sua disponibilidade de realizar as operações de maneira automatizada utilizando o ambiente de programação *VBA*. Caso haja a necessidade de se fatiar o modelo virtual em camadas cujas espessuras não sejam iguais, se tornará necessário realizar uma divisão do modelo virtual inicial em diferentes modelos parciais passíveis de serem fatiados segundo a estratégia de camadas iguais. A questão da execução de superfícies curvas e por conseguinte seu acabamento superficial por exemplo, suscitam a necessidade de uma redução na espessura das camadas.

Essa análise gerou uma rotina desenvolvida na linguagem de programação *Visual Basic* , agregada ao programa desenvolvido neste trabalho que foi denominado de

"AutoCAM", capaz de identificar as dimensões principais do protótipo (nos eixos X, Y e Z) e fatiar em o protótipo virtual em uma quantidade arbitrada pelo usuário de fatias eqüidistantes.

As figuras 3.3, 3.4(a),(b),(c) e (d) exemplificam o processo de posicionamento e realização do fatiamento automatizado no *AutoCAM*.

Como elemento da lógica do programa *AutoCAM*, foram inseridos os conceitos em comando numérico de zero da peça, um ponto de referência no modelo virtual, e origem do sistema de coordenadas, que se trata da referência para a definição de qualquer coordenada dentro do sistema de coordenadas adotado. Para todos os efeitos este ponto é considerado como sendo aquele que possua os menores valores para as coordenadas do modelo nos eixos X e Y, e a maior valor da coordenada no eixo Z em relação à origem do sistema de coordenadas. Assim sendo o *AutoCAM* move a princípio o ponto de referência do modelo para a origem do sistema de coordenadas do espaço virtual.

A FIG. 3.3 elucida a questão da existência do zero da peça , que se trata de um modelo virtual 3D; e sua referência tomada em relação à origem do sistema de coordenadas do espaço virtual, consistindo na adoção do sistema de coordenadas absolutas.

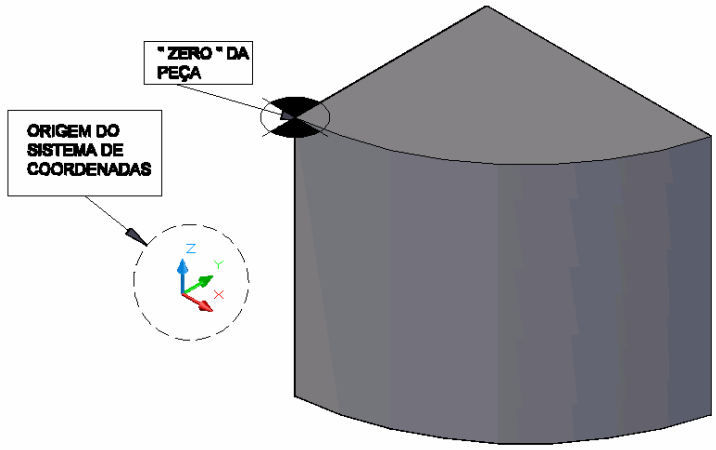


FIG. 3.3- Ilustração acerca das referências adotadas de posicionamento no programa desenvolvido.

O *AutoCAM* também permite através de sua interface, a translação do modelo no espaço tridimensional, que à priori tem seu zero localizado na origem do sistema de coordenadas, para outro ponto qualquer a ser definido pelo usuário.

A escolha das coordenadas do zero da peça (do modelo *CAD*) foi feita considerando-se o método de fabricação e características construtivas da máquina-ferramenta a ser utilizada na fabricação do protótipo. Estes dois fatores implicam que o processo se dará

de cima para baixo no eixo *Z* (no sentido dos valores das coordenadas no eixo *Z* de uma camada posterior serem menores do que as coordenadas *Z* de qualquer camada anterior a esta) para execução do corte no plano *XY*. Com relação aos planos de corte *YZ* e *ZX*, a localização do zero da peça implica no fato de que para a operação do fatiamento, um plano referente a uma camada posterior estará mais distante em relação à origem do sistema de coordenadas bem como do zero da peça, do que qualquer plano referente a qualquer uma das camadas anteriores.

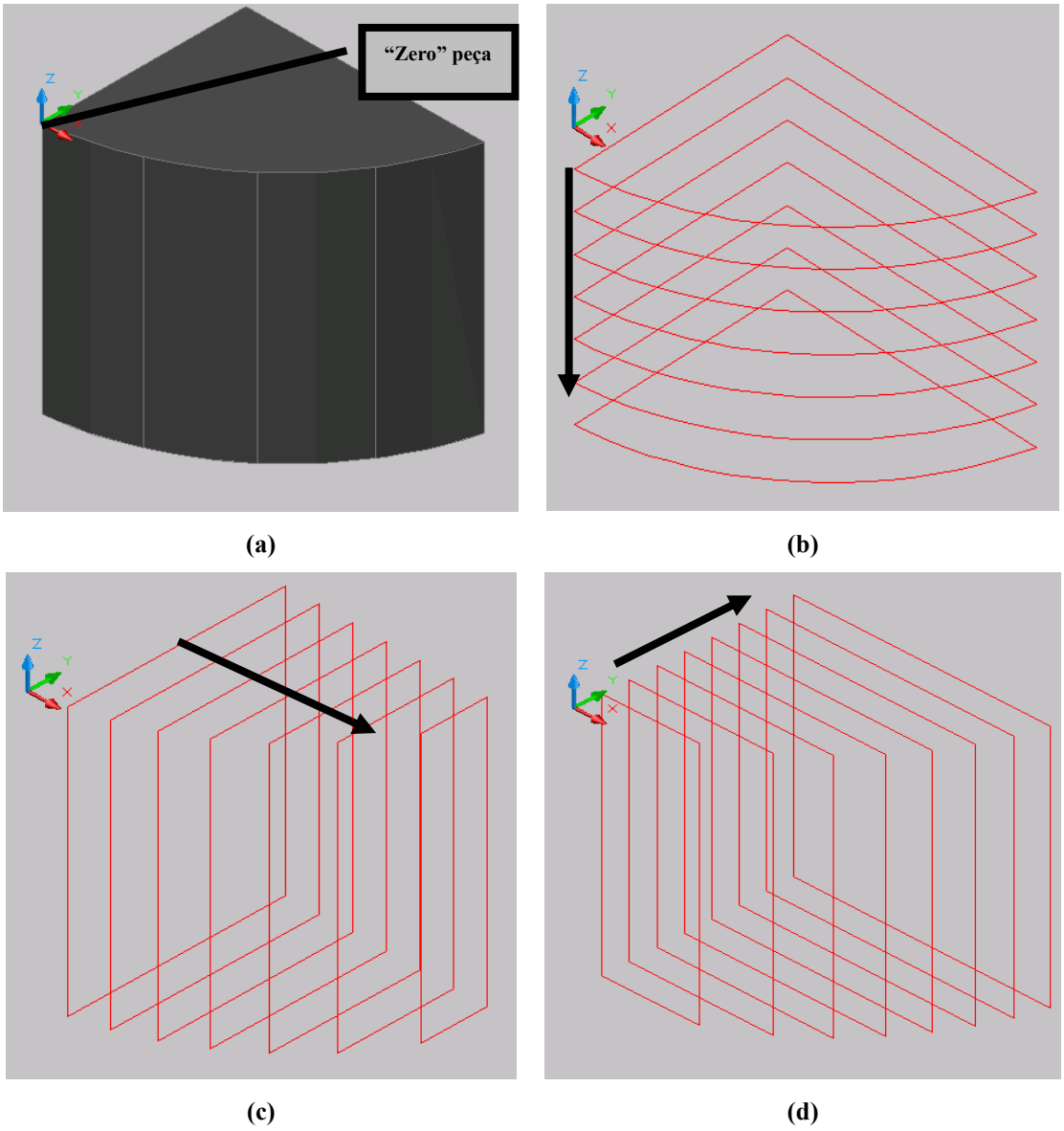


FIG. 3.4- Utilização da rotina de fatiamento implementada no AutoCAM. (a) Exemplo de modelo virtual. (b) Exemplo de fatiamento no plano XY. (c) Exemplo de fatiamento no plano YZ. (d) Exemplo de fatiamento no plano ZX.

Na figura 3.4 estão indicados os sentidos de fatiamento de um modelo *CAD* para os planos *XY* ,FIG. 3.4 (b); *YZ* FIG. 3.4 (c) e *ZX*; FIG. 3.4 (d). O sentido do fatiamento é o mesmo que será empregado na geração do código numérico e portanto será o sentido seguido pela ferramenta de corte no caso da fabricação de um protótipo físico utilizando o código gerado pelo *AutoCAM.*

3.4. Elaboração dos procedimentos de extração, armazenamento e ordenação de dados do *AutoCAM*

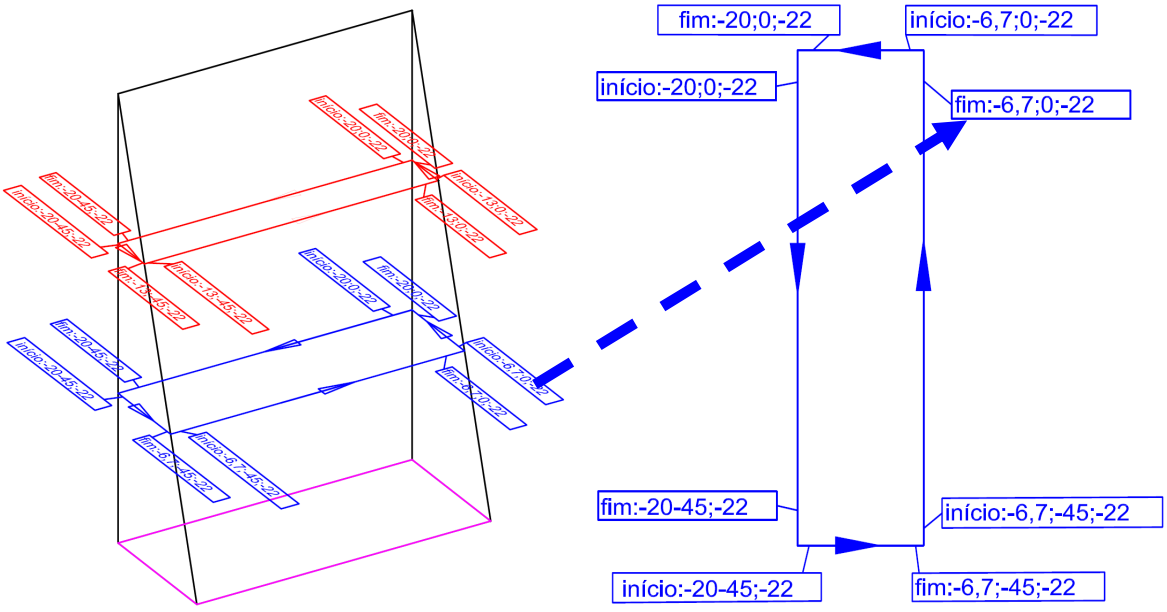
Logo após ser estabelecido o processo de fatiamento o próximo passo, consistiu na pesquisa para realização da extração e armazenamento de dados dos elementos gráficos bidimensionais gerados pelo fatiamento. Em se tratando de dados dos elementos gráficos bidimensionais foi levantada à questão de quais dados seriam relevantes na execução da lógica do programa*,* já levando em consideração a etapa seguinte de ordenação dos dados*.*

Para tanto, realizou-se o estudo sobre o ambiente de programação *Visual Basic* acerca do acesso dos dados aos diferentes componentes do *software* AutoCAD®, o tempo e o momento em que os dados estão disponíveis e para quais componentes do *software* os dados estão disponíveis.

Para o armazenamento dos objetos gráficos foi definida a utilização da alocação dinâmica de memória. Em um primeiro momento uma matriz armazena uma região, produto do fatiamento do modelo em um plano, uma estrutura gráfica bidimensional que é dotada de certas propriedades de interesse. Uma nova variável é constituída dos objetos gráficos bidimensionais, que se tornam acessíveis a partir da "explosão" da região formada por uma fatia. Assim sendo, essa outra variável contém o número exato de camadas definido pelo usuário, armazena todos os objetos gráficos. Todos os que pertencerem à mesma camada estarão na mesma linha. Em uma visão resumida seria uma matriz de dimensões variáveis em que o número de linhas depende do usuário que define a quantidade de camadas, e o número de colunas depende da geometria do modelo virtual na camada correspondente à linha.

Parte da estratégia de ordenação dos dados é devida à observação de como é realizado o armazenamento dos objetos gráficos bidimensionais. Foi evidenciado através de uma série de testes realizados fatiando modelos de geometrias diferentes; que o armazenamento dos objetos gráficos nas variáveis ocorria de modo aleatório (FIG. 3.5). Os objetos formam um contorno fechado, e portanto cada objeto possui um ponto definido por

três coordenadas que é comum a apenas um outro, e; cada objeto possui um ponto de início e um final.



(a) (b)

FIG. 3.5 Ilustração dos processos de extração e ordenação de coordenadas no plano XY.(a) Visualização do processo para todas as camadas. (b) Visualização para uma camada em particular

É válido salientar que o único objeto de interesse, excluído dessa lógica é o círculo que por si só constitui um contorno fechado. Dizer que o armazenamento é aleatório significa então que os objetos cujos pontos são coincidentes não estão em posições consecutivas nas variáveis onde estão armazenados e significa dizer também que não necessariamente o *AutoCAD VBA* considera que se dois objetos possuem um ponto em comum; então um ponto corresponde a o início de um objeto e o fim do outro.

Para o processo de ordenação dos dados elaborado para o *AutoCAM*,o ponto de partida é uma matriz *M* desordenada de tamanho *No x 6* , onde *No* é o número de objetos gráficos identificados na região de corte "explodida". As duas primeiras colunas correspondem às coordenadas do início do objeto e a terceira e quarta coluna correspondem às coordenadas bidimensionais do fim do objeto gráfico. A quinta refere-se à posição do objeto na matriz desordenada e a sexta coluna é reservada para indicação do início e fim do objeto. Deve ser salientado que o processo é passível de ser aplicado a qualquer das categorias consideradas de objetos (linhas, arcos e círculos).

A ordenação é feita a partir dos dados do primeiro objeto gráfico identificado e das coordenadas de seu início e fim considerados. Foi criada uma coluna contendo "ponteiros", que indicam as posições originais dos objetos gráficos, ou seja, que ponto é considerado o início e qual é o fim (FIG. 3.5). Esta etapa é importante para identificar qual a ordem de cada objeto bidimensional e então depois de ordenar, retirar os dados que são de interesse. Das linhas, retiram-se suas coordenadas de início e fim do arco além das coordenadas, retiram-se seu raio e as coordenadas do centro. Já do círculo apenas o raio e as coordenadas do centro são de interesse. Como parte da ordenação foi introduzido um *flag* (elemento indicador) para acusar a inversão de sentido (necessidade de troca do início do elemento gráfico com o seu fim).

Em uma próxima etapa os objetos são reordenados de modo que o elemento mais próximo do limite inferior da região "cortada" para "cortes" no plano *XY*. Para os outros planos o ponto escolhido tem uma das coordenadas *X* e Y mínimas e em *Z* máxima. Os pontos de limite são obtidos pela propriedade *getBoundingBox* (AUTODESK,2004) do *AutoCAD VBA*.

Na ordenação não importa se a trajetória é feita no sentido horário ou anti-horário, isso vai depender basicamente de qual será o primeiro objeto gráfico mais próximo, e qual seu sentido.

A questão do sentido de execução da trajetória seria interesse se o programa trabalhasse com modelos com cavidades. O fatiamento de modelos com cavidades gerariam mais de uma região, ou seja, mais de um contorno fechado. Estas regiões implicam em termos práticos em conjuntos de áreas que com "preenchidas de material" e "vazios" que deveriam ser identificadas conforme de fato ocorre na geração de trajetória para a prototipagem rápida. Esta questão da chamada "profundidade" das superfícies, que são os seus espaços "cheios" e "vazios" é tratada por TATA (1998), QIU, LANGRANA, DANFORTH, SAFARI, JAFARI (2001) , YANG (2002) e por CHOI, KWOK (2004). Análise da literatura pesquisada sobre este assunto leva ao raciocínio de que dentro dessa lógica caberiam infinitas regiões "cheias de material" e com "vazios", umas dentro das outras (FIG.3.6). Desta maneira todo o processo de ordenação realizado neste trabalho deveria ser aplicado a todos os contornos fechados. O programa também seria obrigado a identificar que região é "cheia" e qual é "vazia" de modo a efetuar de maneira correta uma possível compensação de raio da ferramenta. Tal operação para ser realizada de maneira sistemática e automatizada envolveria uma lógica complexa que poderia inviabilizar a conclusão do programa computacional em tempo hábil.

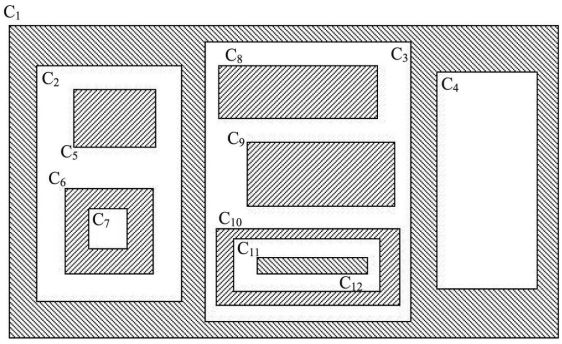


FIG. 3.6 - Ilustração das infindáveis possibilidades de contornos "cheios" e "vazios" uns dentro dos outros – Fonte - CHOI, KWOK (2004)

Para a prototipagem rápida esta preocupação com os contornos e o sentido da trajetória que é empregado na identificação do que é "cheio" e o que é "vazio" faz sentido pois o material é depositado nas regiões "cheias" como ilustrado por meio de hachuras na figura 3.6. O programa desenvolvido é voltado para prototipagem por retirada de material. A estratégia adotada torna irrelevante ao programa o processo de identificação e transfere ao usuário a responsabilidade de saber qual região é "cheia" e qual é a "vazia". Outro argumento para não adotar esta filosofia de trabalhar com contornos internos uns aos outros é que em termos de retirada de material, é mais lógico e prático se fazer cada cavidade completa na peça do que fazer de modo gradativo todas cavidades presentes em uma camada.

Mesmo sem a sofisticação de lidar com contornos internos uns aos outros, será observado mais adiante que é possível introduzir cavidades no protótipo utilizando o *AutoCAM*.

Um dos objetivos da estratégia de ordenação é evitar a colisão indesejada da ferramenta com a peça durante sua translação. Isso é realizado estabelecendo um ponto zero em cada camada, cujas coordenadas absolutas foram definidas. É observável que entre uma camada e outra igualmente espaçada não há uma variação brusca de geometria. Considerando ainda que o ponto final de uma região fechada coincide com o início do mesmo; pode-se afirmar que houve uma otimização da trajetória entre uma camada e outra.Para ser mais claro, como se trata de um objeto contínuo, é de se esperar que as seções adjacentes não sejam muito diferentes umas das outras, e portanto, a distância de um vértice ao ponto estabelecido (o zero do modelo 3D) será a mais próxima (podendo ser até coincidente dependendo da geometria e orientação do modelo).

Outro objetivo seria a otimização da trajetória de fabricação do protótipo, que seria a redução do tempo dentro do qual a ferramenta desloca mas não executa o corte.

Analisando a execução da trajetória no plano XY, por exemplo. É notável que, sendo utilizado o limite inferior do contorno fechado; não estirão objetos cujas coordenadas nos planos X e Y sejam menores que esta. O caso mais extremo seria o da coincidência das coordenadas nos planos X e Y. Por conseguinte a trajetória linear do translado da ferramenta do ponto mínimo ao início da trajetória de corte na camada é mínima ou nula e não transpassa nenhum objeto gráfico do contorno fechado.

A figura 3.7 auxilia o entendimento tanto da ordenação quanto da otimização sustentada pelos argumentos utilizados.

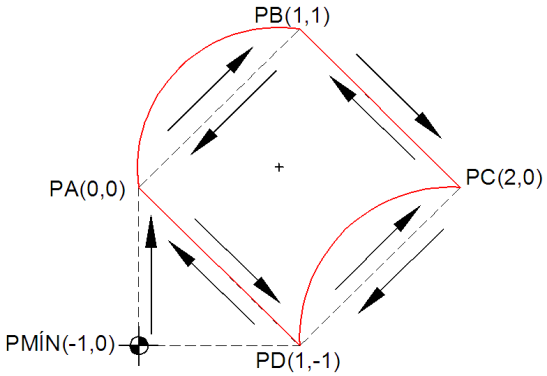


FIG. 3.7- Ilustração do processo de ordenação para uma camada no plano XY

3.5- A visualização do modelo CAD utilizando o *AutoCAM*

A princípio procurou-se utilizar a mesma estratégia de visualização que foi empregada no *3DFORM 2.0.* A referida estratégia consiste na manipulação também utilizando a tecnologia *ActiveX* de um *software* visualizador que é inserido em um formulário do *3DFORM 2.0.* Desta maneira, este *software* visualizador passa a ser parte integrante do *3DFORM 2.0.*

O visualizador em questão é o *VoloView*® , também empresa *AutoDesk*™. Este produto fez parte do pacote de recursos até a versão *AutoCAD*® 2004 (R. 16.0), sendo substituído por outro *software* (*AutoDesk*™ *DWF*™ *Viewer*) para versões superiores.O *VoloView*® poderia até ser utilizado ainda como visualizador para o programa desenvolvido, porém considera-se que não há garantias que este *software* de visualização possa ser aplicável em versões posteriores do *AutoCAD*® como a utilizada no presente trabalho. Também não há garantias que o mesmo seja capaz de utilizar os novos recursos agregados a cada atualização do *AutoCAD*®, uma vez que o *VoloView*® não é mais distribuído conjuntamente com o *AutoCAD*®. Com relação ao novo software visualizador, o *AutoDesk*™ *DWF*™ *Viewer,* foi

concebido com o intuito de ser empregado na distribuição de arquivos na *Internet* e ao que parece não tem este viés de ser agregado à tecnologia de automação *ActiveX*, o que também o descarta sua utilização como visualizador. Para lidar com esta restrição com relação à visualização do modelo *CAD*, foi feita a opção de utilizar o visualizador do próprio *AutoCAD*®, ou seja, a própria tela de interface do *AutoCAD*®. Além disso, foi observado um benefício em utilizar a própria tela do *AutoCAD*®,uma vez que se torna possível selecionar o modelo 3D diretamente na tela do programa *CAD*. Assim, um dos recursos desejados para o *AutoCAM*, que é a correção de raio da ferramenta na geração das trajetórias é feita no modelo 3D, alterando suas dimensões. Este processo envolve a seleção das faces do sólido a sofrerem a correção, que por sua vez podem exigir a movimentação e rotação do modelo 3D, obtendo-se então o retorno em termos de visualização da alteração do modelo 3D. A realização de tal sequência de operações é assegura com a estratégia adotada, e utilizando um visualizador poderia não ser viável por causa dos recursos oferecidos pela tecnologia de automação voltada para as tarefas desta seqüência.

3.6- Funcionamento e interface do *AutoCAM*

O fluxograma abaixo representado pela figura 3.8 descreve o modo como o AutoCAM opera em sua tarefa de gerar código CNC. Estão contempladas no fluxograma as entradas e saídas de cada procedimento realizado e o modo como os dados interagem e em que momento eles se fazem necessários segundo sua lógica.

A descrição realizada no fluxograma serve de guia ao usuário , uma vez que ele define a ordem de execução dos procedimentos realizáveis pelo programa *AutoCAM*.

A apresentação a seguir da interface do *AutoCAM* e o fluxograma apresentado colaboram no entendimento de como o *AutoCAM* foi concebido,qual sua é a sua função, de que maneira ele realiza sua função e por fim, como deve ser utilizado para seu correto funcionamento. A interface do *AutoCAM* se apresenta inicialmente conforme a figura 3.9

O balão 1 da figura 3.9, aponta para o ícone criado para o *AutoCAM* dentro da interface *AutoCAD*®. Ao clicar neste ícone, aciona-se um *Macro* responsável pela inicialização do arquivo do programa *AutoCAM*. Sua interface é apresentada na tela à esquerda da figura 3.9. Obviamente a tela à direita se trata da tela do *AutoCAD*®.

O *AutoCAM* foi desenvolvido e é aplicado no ambiente do *AutoCAD VBA*, inserido no *AutoCAD*®. Ele existe sob a forma de um arquivo de projeto do *AutoCAD VBA* cuja extensão de arquivo é *.DVB*. De sorte que por apresentar-se desta maneira, o *AutoCAM*

não é um programa executável, ou seja, ele precisa de uma plataforma para sua execução, que é fornecida pelo *AutoCAD VBA*. O ambiente de programação *AutoCAD VBA*, é uma versão resumida e voltada à aplicação para o *AutoCAD*®, do ambiente de programação *VB*, e portanto não apresenta o recurso do *Visual Basic*®, de gerar programas executáveis. Para tanto o programa deveria ter sido desenvolvido utilizando o ambiente de programação do *Visual Basic*®.

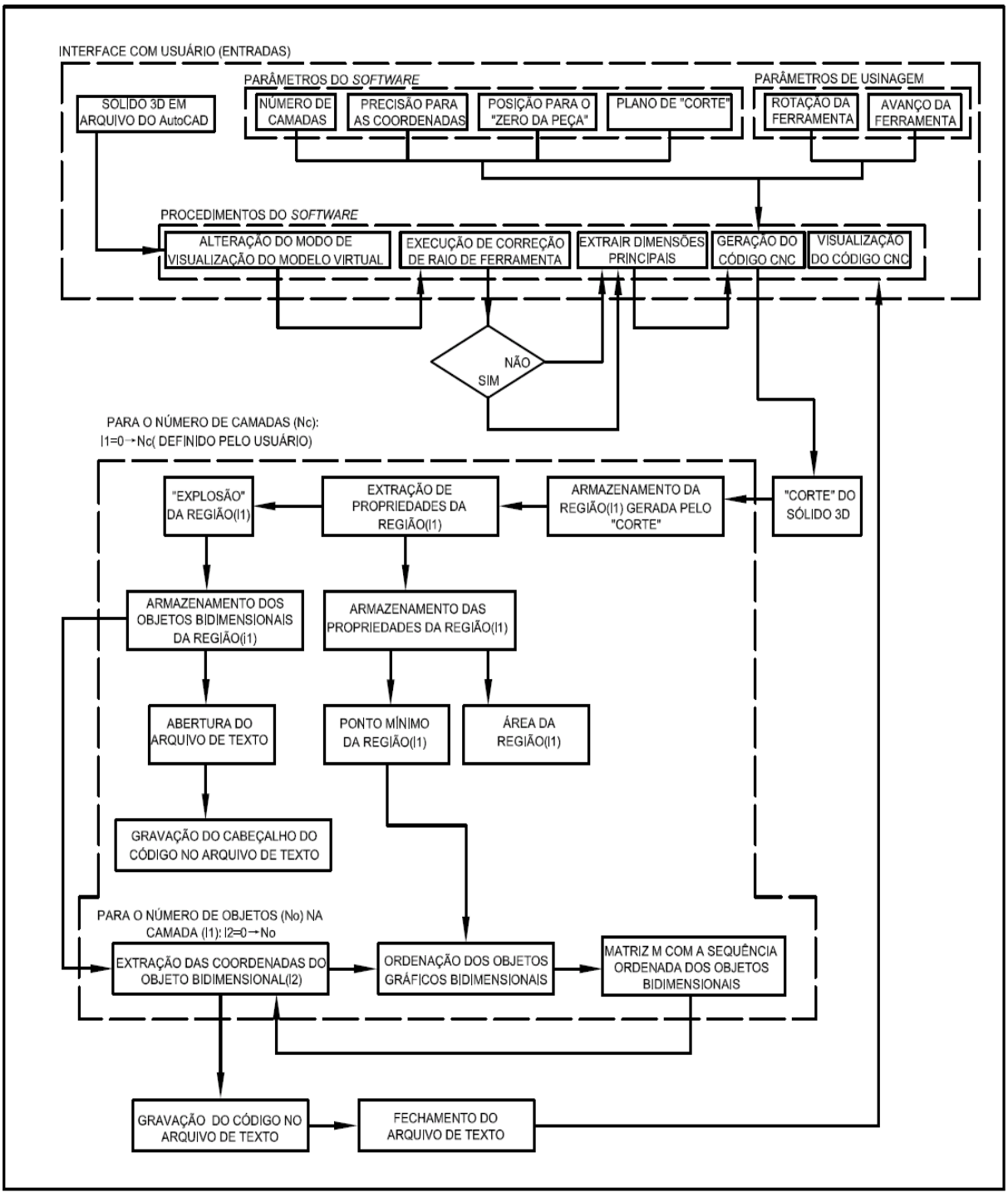


FIG. 3.8-Fluxograma descritivo do *AutoCAM*

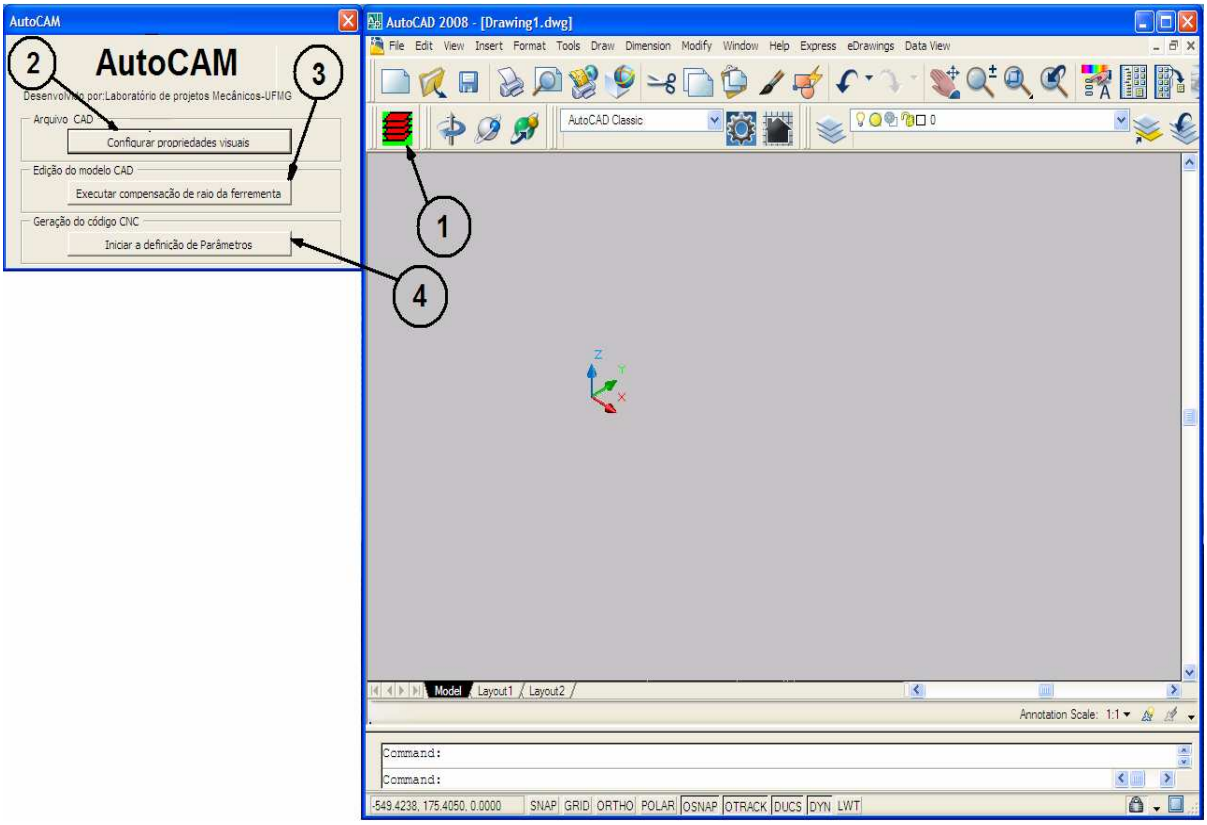


FIG. 3.9-Interface inicial do *AutoCAM*

Em termos de programação, migrar para o ambiente *Visual Basic*®, não acarretaria em grandes mudanças, pois a linguagem utilizada é a mesma, o *VB*. Mas também não quer dizer que seja absolutamente igual. O *AutoCAD VBA* apresenta algumas funções mais "doutrinadas" ao trabalho com *AutoCAD*®, o que a levaria a necessidade de ligeiros ajustes no código fonte do programa.

Em se tratando da execução do programa, embora o mesmo não tenha sido desenvolvido no ambiente de programação *VB,* forma realizados testes com partes do código fonte do *AutoCAM* adaptadas para este ambiente de programação. Destes testes, pode-se observar que a execução do programa se torna mais lenta. Este fato vem a pesar contra a migração para o ambiente (ou plataforma) de programação do *Visual Basic*® mesmo que por sua vez, este possua o atrativo da geração de um programa executável.

O balão 2 da figura 3.9, aponta para o primeiro recurso do *AutoCAM* a ser após a abertura do arquivo *CAD*. Abertura que podes ser realizada tanto antes quanto após a inicialização do programa, sem prejuízo nenhum ao funcionamento. Este recurso é o da manipulação das propriedades visuais. Uma vez que tenha sido carregado um arquivo *CAD*, utilizando a interface do *AutoCAD*® . o primeiro recurso chamado de "configurar propriedades

visuais", aplica ao modelo um dos modos de visualização oferecidos pelo *AutoCAD*®, o modo *Realistc* e o apresenta em uma das vistas padronizadas, no caso uma vista isométrica chamada *SE Isometric.*

O balão 2 da figura 3.9, aponta para o segundo recurso do *AutoCAM* a ser após a abertura do arquivo *CAD*. Este é um recurso opcional, o da aplicação da compensação do raio da ferramenta feito através da correção de faces do modelo. Trata-se de um recurso opcional porque o modelo pode ser projetado já com suas medidas corrigidas.

A clicar no referido botão 2, uma nova tela da interface irá surgir (FIG. 3.10). Esta interface irá explicar ao usuário do *AutoCAM* como este deve proceder para realizar com sucesso a correção no modelo.

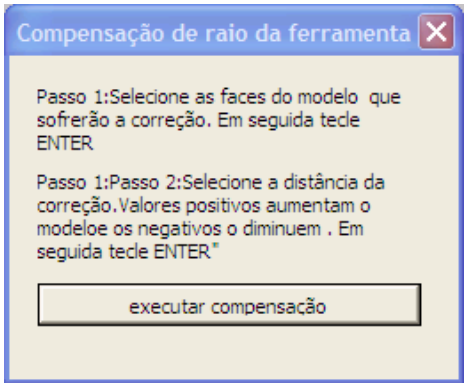


FIG. 3.10-Interface do *AutoCAM* referente ao processo de compensação de raio da ferramenta.

Após passar pelos procedimentos indicados pelos balões 1,2 e 3 da figura 3.9, o usuário do *AutoCAM* deve clicar no botão referente ao quarto procedimento, indicado pelo balão 4 também da figura 3.9. Esta ação irá abrir uma nova interface (FIG. 3.11) dentro da qual serão escolhidos e introduzidos parâmetros para então culminar na geração do código numérico.

Na utilização desta interface (FIG. 3.11), o primeiro passo é extrair as dimensões, o que é de modo intuitivo realizado através do botão "extrair dimensões". Esta prática fornece dados ao programa utilizados na definição do fatiamento do modelo além de retornarem ao usuário as dimensões principais também necessárias a ele para definição do número de camadas e por conseguinte da espessura das mesmas.

Em seguida o usuário do *AutoCAM* deve mover o zero do modelo para o ponto desejado no sistema de coordenadas do espaço virtual. A princípio, ao clicar apenas no botão

"alterar coordenadas" o usuário posiciona o zero do modelo na origem do sistema de coordenadas.

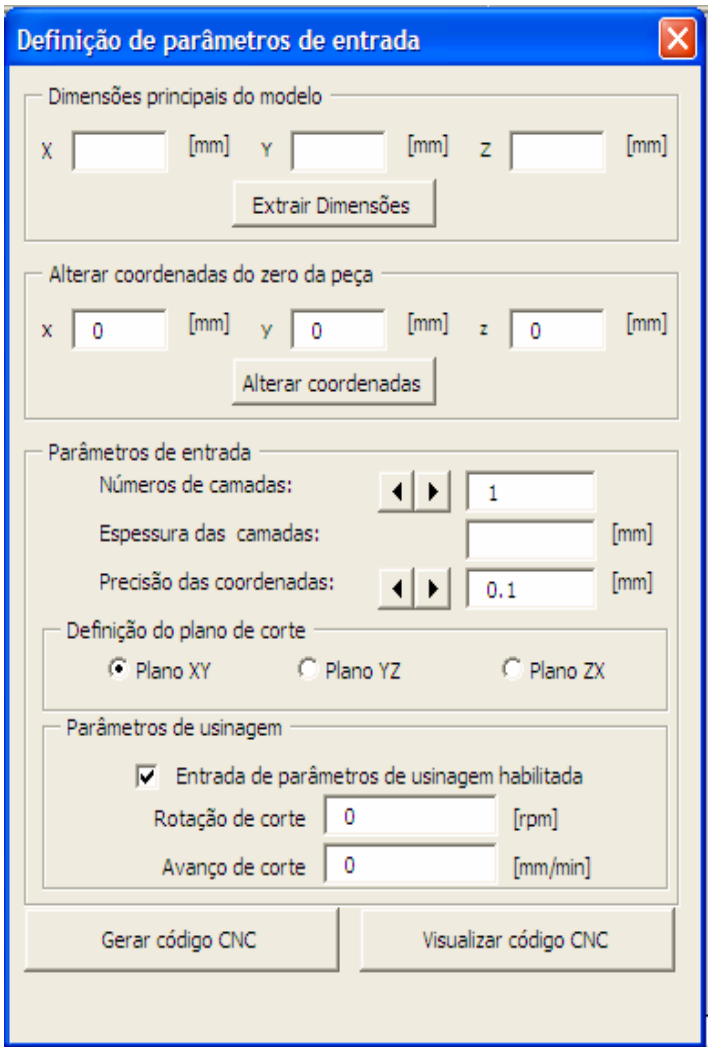


FIG. 3.11-Interface do *AutoCAM* referente à definição de parâmetros e geração do código numérico.

Em uma próxima etapa o usuário seleciona o plano de corte (fatiamento) desejado e então define os parâmetros para a geração do código. Basicamente o usuário definirá o número de camadas com as quais o modelo será fatiado no plano escolhido e o número de casas decimais a ser utilizado nas coordenadas presentes no código numérico a ser gerado.

Existe também logo abaixo nesta mesma interface (FIG. 3.10) um espaço para definição opcional de parâmetros básicos de usinagem.

Tendo definidos todos os parâmetros resta ao usuário ordenar ao *AutoCAM* a geração do código numérico, que é feito através do botão "gerar código CNC". Quando o programa terminar a geração ele retornará uma mensagem ao usuário. Depois desta mensagem o arquivo estará disponível ao usuário sob a forma de texto.

3.7- Validação do código CNC gerado

Foram realizados testes no programa para verificar a conformidade do código gerado com a geometria do modelo virtual e para verificar a implementação das estratégias agregadas ao *software* como o processo de geração de trajetória e definição do zero da peça (do modelo 3D que representa a trajetória de usinagem) translação do zero da peça no sistema de coordenadas do espaço virtual; nos três planos possíveis de fatiamento.

A visualização da trajetória utilizada na validação foi feita através do *software CNC Simulator* (FIG. 3.12). Os códigos *CNC* gerados são as entradas necessárias para o *CNC Simulator*. A partir do código e da definição de alguns parâmetros como as dimensões de um bloco virtual no qual ele promove a visualização da evolução da usinagem.

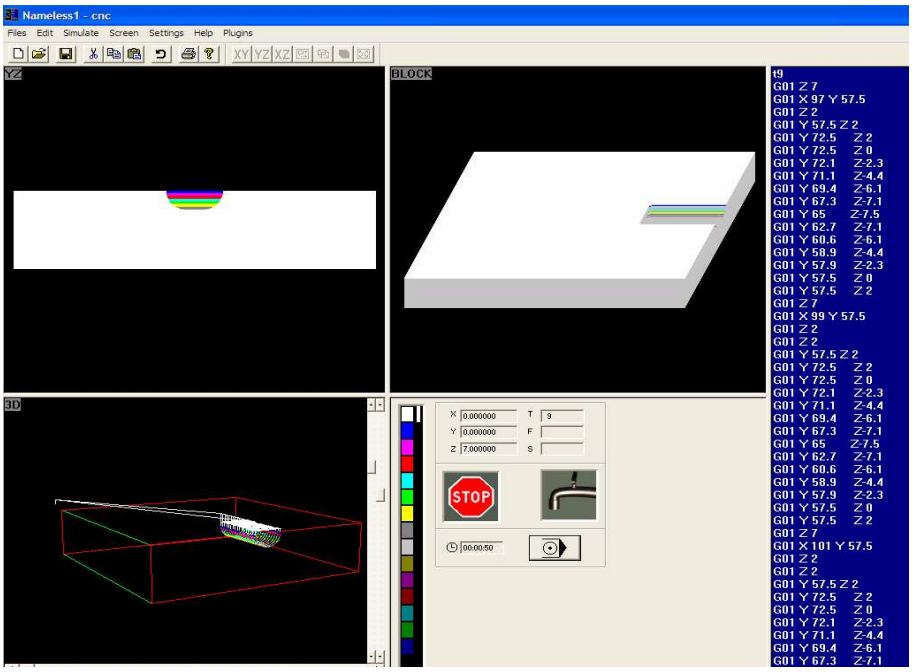


FIG. 3.12-Ilustração da validação do código gerado pelo *AutoCAM* utilizando o *CNC Simulator*

3.8. Estratégia de utilização do *AutoCAM* para prototipagem

O programa computacional *AutoCAM* foi idealizado em principio para se realizar a prototipagem por retirada de material, considerando não tem uma estratégia pré-definida para preenchimento de contornos fechados. A utilização do *AutoCAM* é baseada em considerar que o sólido virtual modelado em *CAD* representa a trajetória de usinagem. Ao ser fatiado em diversas camadas, surgem diversos contornos fechados. Após a realização da

usinagem, a peça final corresponde à somatória da parte interna aos contornos fechados de todas as camadas.

A estratégia para conseguir usinar uma peça com cavidades utilizando o *AutoCAM* seria utilizar quantos modelos *CAD* forem necessários para representar geometria da cavidade. Ao posicionar o modelo 3D que representa a cavidade dentro de um contorno fechado de um outro modelo 3D (FIG.3.13(a)) , então é inserida uma cavidade neste último (FIG.3.13(b)). A interseção de um modelo 3D com o contorno externo de outro gera então uma modificação na geometria da peça a ser usinada (Figuras 3.13(c),(d)).

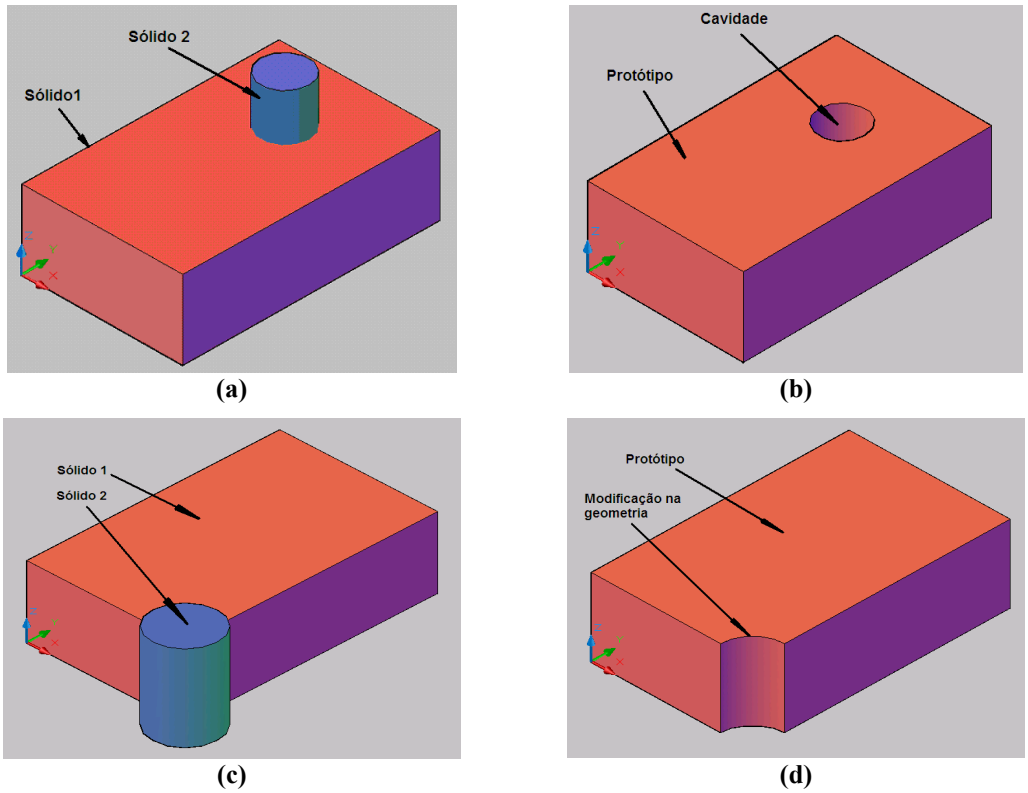


FIG. 3.13(a),(b),(c),(d)- Ilustração da estratégia de fabricação de protótipos utilizando o *AutoCAM*

Em nenhum momento durante a utilização do programa ocorre de se trabalhar com mais de um modelo virtual simultaneamente como ilustrado nas figuras FIG. 3.13(a) e (c). Os códigos são gerados separadamente para cada um dos sólidos e as operações de interseção ocorrerão diretamente no protótipo físico na execução dos códigos gerados. O usuário do *AutoCAM* deve utilizar a estratégia para execução do protótipo de gerar os sólidos necessários para descrever as geometrias desejadas, definindo a posição relativa entre eles, definindo suas coordenadas globais em relação à origem do sistema de coordenadas do espaço virtual do *AutoCAD*; e por fim definindo a seqüência de realização dos códigos gerados.

4. RESULTADOS

4.1 Fabricação do protótipo físico

Um modelo 3D foi idealizado, visando explorar o potencial de aplicação do programa AutoCAM através da geração do código de comando numérico e implementação da estratégia para obtenção do protótipo físico .A Figura 4.1 apresenta o modelo 3D idealizado e na figura 4.2 algumas de suas dimensões podem ser observadas.

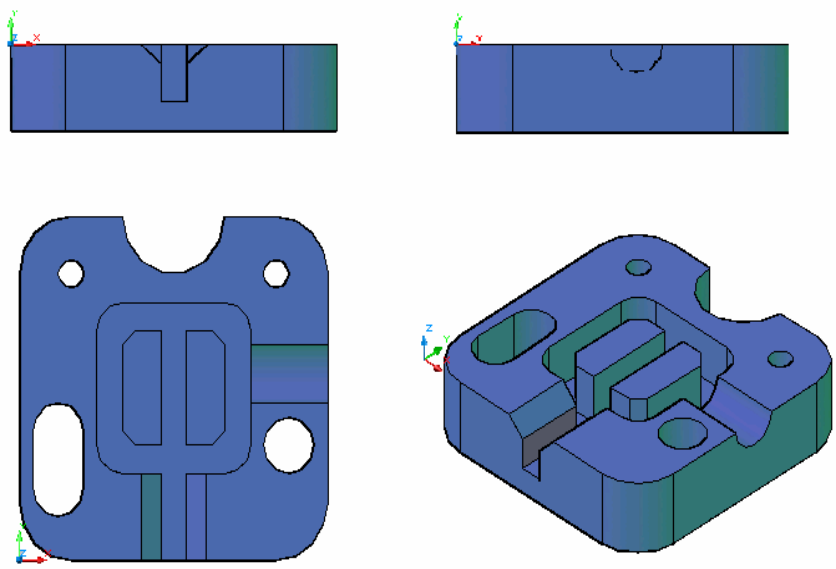


FIG. 4.1– Modelo CAD do protótipo idealizado para aplicação do AutoCAM na prototipagem por retirada de material .

O modelo virtual do protótipo físico foi desmembrado em 13 (treze) modelos parciais diferentes a serem trabalhados separadamente. A geometria de cada modelo parcial foi elaborada de modo a se obter as características geométricas projetadas para o protótipo físico considerando-se a realização da correção do raio da ferramenta quando se fizer necessário.

Os modelos que representam as trajetórias já poderiam ter sido construídos com as medidas corrigidas, porém no intuito de aplicar o recurso de compensação de raio da ferramenta oferecido pelo programa *AutoCAM*, foram utilizadas as dimensões exatas das cavidades, rebaixos e reentrâncias existentes no protótipo virtual modelado em CAD.

Para melhor compreensão da estratégia, os modelos foram numerados como mostra a figura 4.3, na ordem que a priori deveria ser seguida para a aplicação do código CNC gerado na fabricação do protótipo físico. Também no intuito de esclarecer a estratégia adotada de aplicação do programa, está indicado próximo de cada balão de identificação do modelo, o plano de usinagem de sua geometria.

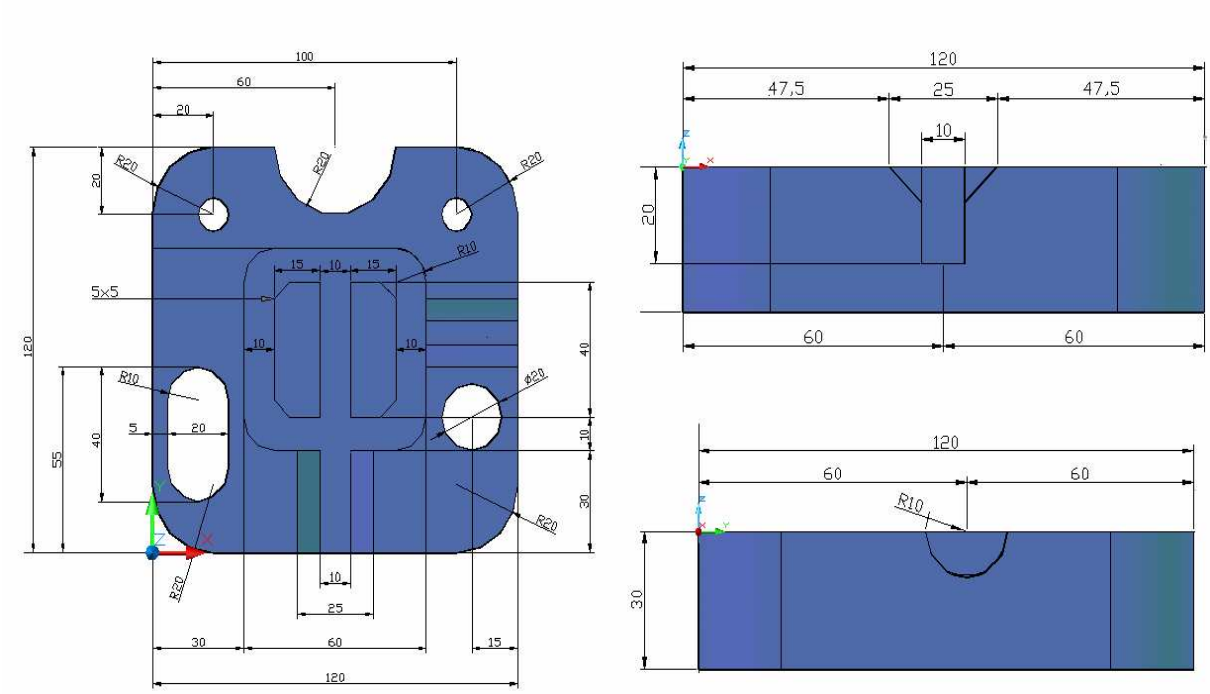


FIG. 4.2– Dimensões do protótipo idealizado.

Como cada modelo parcial é trabalhado de maneira individual no *AutoCAM*, é necessário que o usuário tenha definido previamente as distâncias relativas entre cada modelo parcial. Porém todas as coordenadas a serem definidas no programa devem ser em relação à origem do sistema de coordenadas do espaço virtual do *AutoCAD.* A outra referência utilizada para estabelecer as distâncias relativas utiliza o princípio de definição do zero da peça.

Considerado como sendo o ponto mínimo nas coordenas nos eixos *X*,*Y* e *Z* , ou seja das coordenadas dos vértices (ou de suas projeções) mais próximos da origem do sistema de coordenadas do espaço virtual, para usinagem no plano XY. Para usinagem nos planos *YZ* e *ZX*, foi tomada a coordenada máxima e no eixo *Z*. É em relação a este ponto que as coordenadas devem ser definidas. Na figura 4.4 podem ser vistas os zeros de cada modelo, bem como a localização da origem do sistema de coordenadas.

Nas figuras 4.3 e 4.4 podem ser visualizadas sob a forma de linhas tracejadas, as projeções dos contornos das geometrias desejadas no protótipo virtual para demonstrar a

necessidade da correção do raio da ferramenta executando alterações na geometria dos modelos referentes às trajetórias de usinagem.

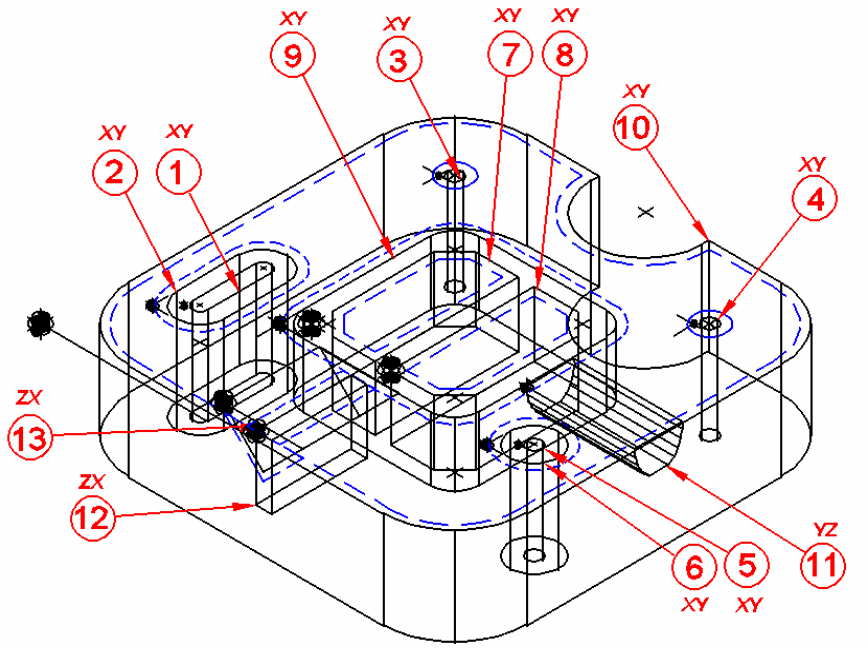


FIG.4.3- Identificação dos modelos referentes às trajetórias de usinagem utilizados na geração do código CNC e dos respectivos planos de usinagem.

Uma vez feito o delineamento da estratégia de aplicação do programa para o caso deste protótipo, em uma próxima etapa os modelos virtuais são gerados separadamente e também armazenados em arquivos *CAD* diferentes para então utilizar o *AutoCAM*. O próprio *AutoCAD* foi utilizado na modelagem dos modelos 3D por uma questão de conveniência; porém como já foi tratado anteriormente, nada impede ao usuário do *AutoCAM* de modelar cada parte em outro *software CAD*. Logicamente que este outro *software CAD* deve possuir recursos de converter seu arquivo em algum dos formatos com os quais o *AutoCAD* trabalhe.

Na figura 4.5, 10 modelos parciais utilizados podem ser vistos. São em número de dez, porque existem compatibilidades entre as geometrias de alguns modelos parciais. Os modelos parciais 1 e 2 (FIG. 4.3) são utilizados para fazer o furo oblongo em duas etapas, como pode ser visto no canto inferior esquerdo da vista no plano *XY* do modelo 3D na figura 4.4 .Da mesma maneira os modelos 5 e 6 (FIG. 4.3) são utilizados na obtenção do furo de diâmetro maior, visto no canto inferior direito da vista no plano *XY* do modelo 3D na figura 4.3. O último caso ocorre entre os furos menores representados pelos modelos 3 e 4 (FIG. 4.3) que possuem as mesmas dimensões porém estão em posições diferentes.

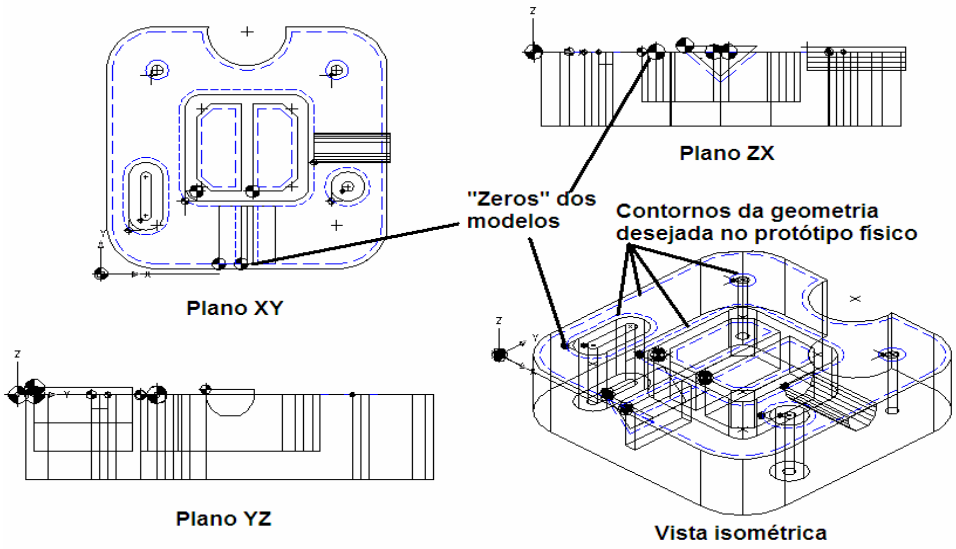


FIG.4.4- Visualização dos zeros dos modelos 3D, origem do sistema de coordenadas do espaço virtual e as projeções dos contornos desejados no protótipo físico .

Embora os modelos 7 e 8 (FIG. 4.3) possuam as mesmas dimensões, eles possuem orientações diferentes e o *AutoCAM*, em princípio, não dispõe de recurso de rotação automática do modelo em torno dos eixos de coordenadas (embora possa ser realizada esta tarefa na manipulação direta do *AutoCAD*). Desta maneira os dois modelos não podem ser resumidos e um único modelo.

Após a realização de correção de raio interna (no sentido de diminuir as dimensões do modelo) de 7,5 mm utilizando o *AutoCAM*; o modelo A (FIG. 4.4) corresponde ao modelo 1 (FIG 4.3). O mesmo modelo A após uma correção de raio interna de 2,5mm, corresponde ao modelo 2 (FIG 4.3).

O modelo B (FIG. 4.5) após uma correção de raio interna de 2,5mm e definição das duas diferentes localizações para seu zero, corresponde aos modelos 3 e 4 da figura 4.3.

O modelo C (FIG. 4.5) de maneira análoga ao modelo A (FIG. 4.5), sofre duas correções. Em uma primeira correção de raio interna de 7,5mm, este modelo passa a corresponder ao modelo 5 (FIG. 4.3). Na segunda correção de raio interna, agora de 2,5mm, este modelo passa a corresponder ao modelo 6 (FIG. 4.3).

Já os modelos D,E e G (FIG. 4.5), após a efetuação de correções externas (no sentido de aumentar as dimensões do modelo) de 2,5mm passam a corresponder respectivamente aos modelos 7, 8 e 10 (FIG. 4.3).

O modelo I após uma correção interna de 2,5mm corresponde ao modelo 11 (FIG. 4.3).

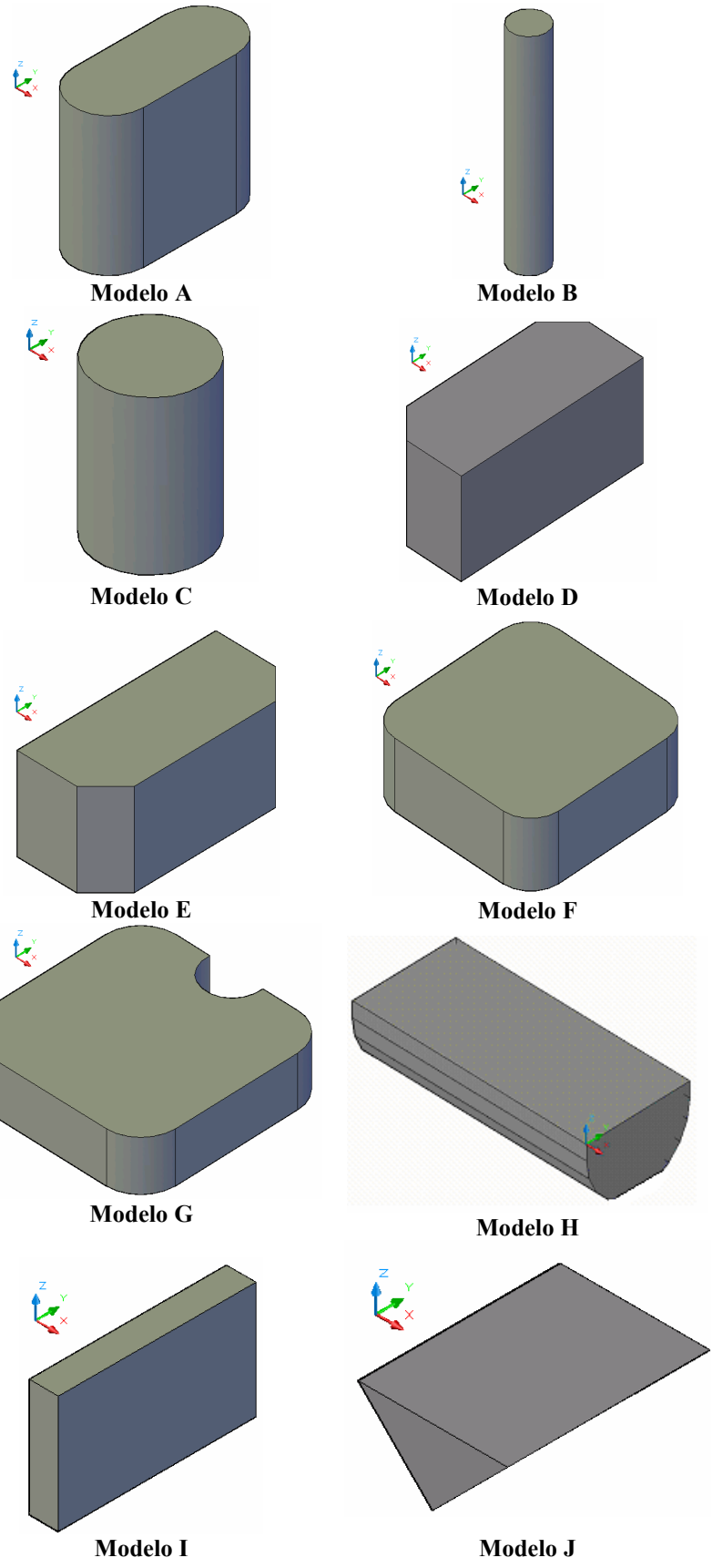


FIG.4.5-Modelos utilizados como entradas para o AutoCAM, que após a correção correspondem os modelos numerados de 1 a 13 (FIG. 4.3).

Devido à sua geometria e o plano de usinagem escolhido, o plano *ZX* (FIG. 4.3), o modelo J (FIG. 4.5). foi o único a não sofrer a correção em suas dimensões. A correção foi realizada deslocando o modelo que corresponde cunha triangular 3,5mm no sentido positivo, no eixo *Z* de coordenadas.

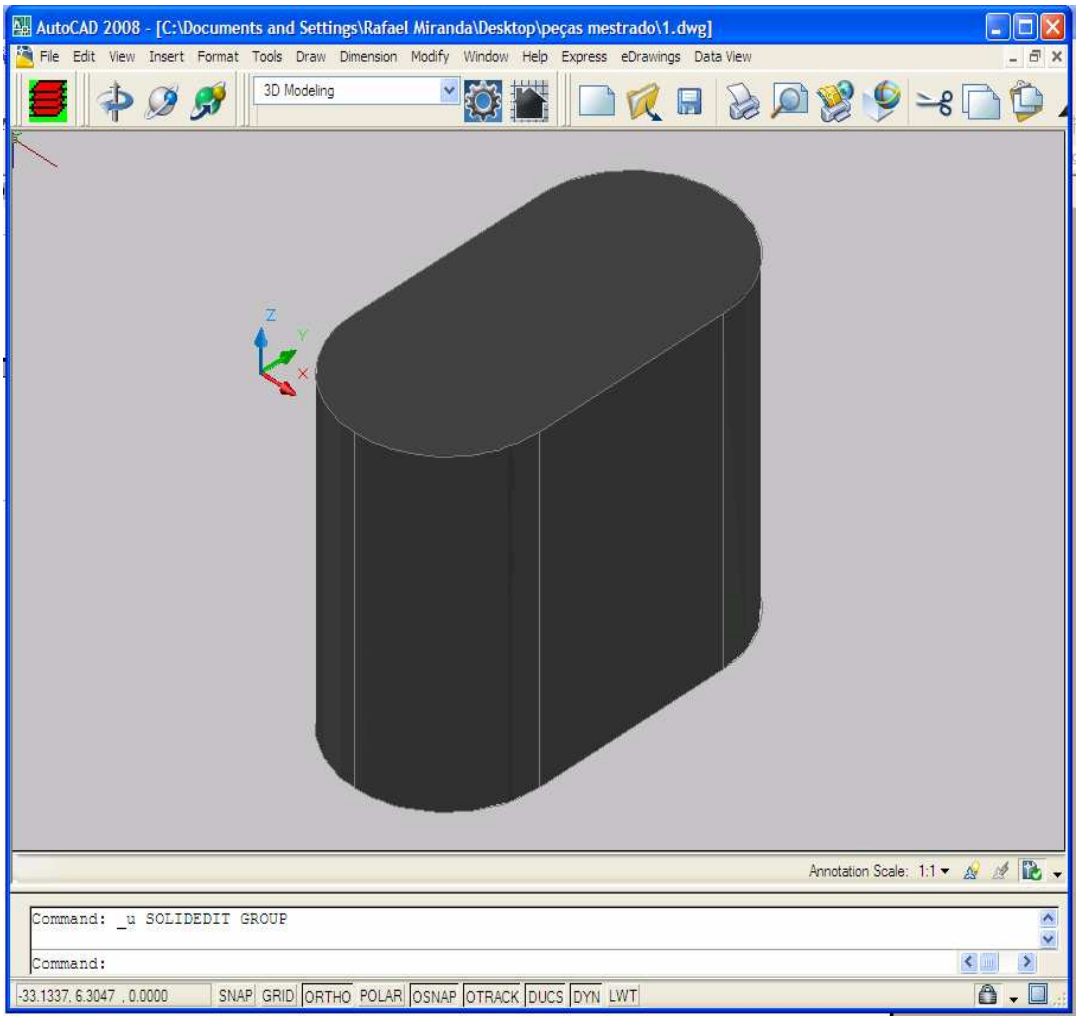


FIG.4.6-Abertura do arquivo CAD do modelo A (FIG. 4.5)

A seguir será apresentada uma seqüência ações que ilustram diferentes etapas de utilização do programa *AutoCAM*, desde a abertura do arquivo *CAD* (FIG. 4.6), contempla a compensação de raio da ferramenta no modelo (FIG. 4.7 e 4.8), a definição de parâmetros (FIG. 4.9) e finalizando com a geração e visualização do código numérico. Será utilizado o modelo A (FIG. 4.5), de modo a se obter a trajetória de usinagem sob a forma de código numérico, o código G; referente ao modelo 1 (FIG. 4.3).

Os códigos *CNC* gerados e a visualização das trajetórias utilizadas através do *software CNC Simulator* são mostrados no Anexo II.

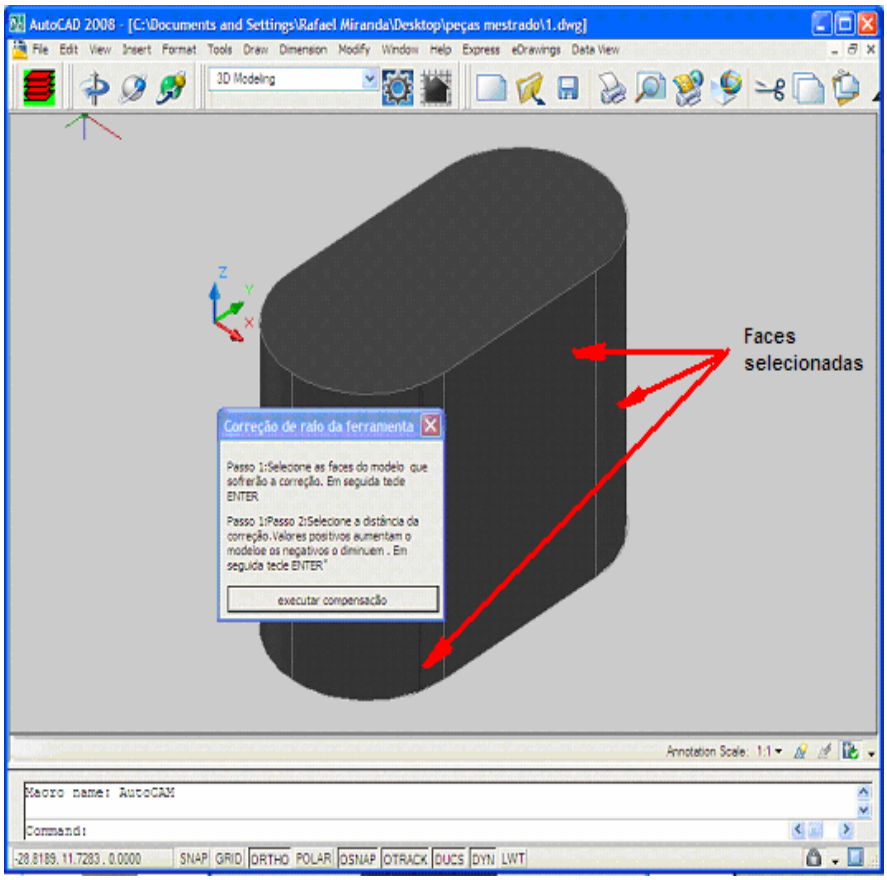


FIG.4.7-Recurso de edição do modelo A (FIG. 4.5), correção do raio da ferramenta.

Na figura FIG 4.7 existem indicações de algumas das faces selecionadas para a correção. Trata-se de uma operação de muita cautela, pois apenas as faces do contorno a ser corrigido devem ser selecionadas. Selecionando por exemplo uma aresta ou a face do topo do modelo A (FIG 4.5), acarreta na execução da correção na direção do eixo *Z* que não é desejada.

Após a definição da localização do modelo e definição dos parâmetros: plano de corte, número de camadas, precisão para as coordenadas (número de casas decimais utilizadas na representação dos valores da coordenadas no código CNC gerado) e parâmetros de usinagem; o usuário do *AutoCAM* deve prosseguir para a geração do código de comando numérico (QUADRO 4.1). Na figura 4.10, pode ser visto o modelo já fatiado após o processo de geração do código numérico.

De posse do código *CNC* gerado pelo *AutoCAM*, prossegue-se à verificação da trajetória de usinagem contida em tal código. A verificação é realizada utilizando o *software CNC Simulator*. Na figura FIG 4.11, pode ser vista a verificação para o caso do modelo 1 (FIG. 4.3) que vem sendo trabalhado até então.

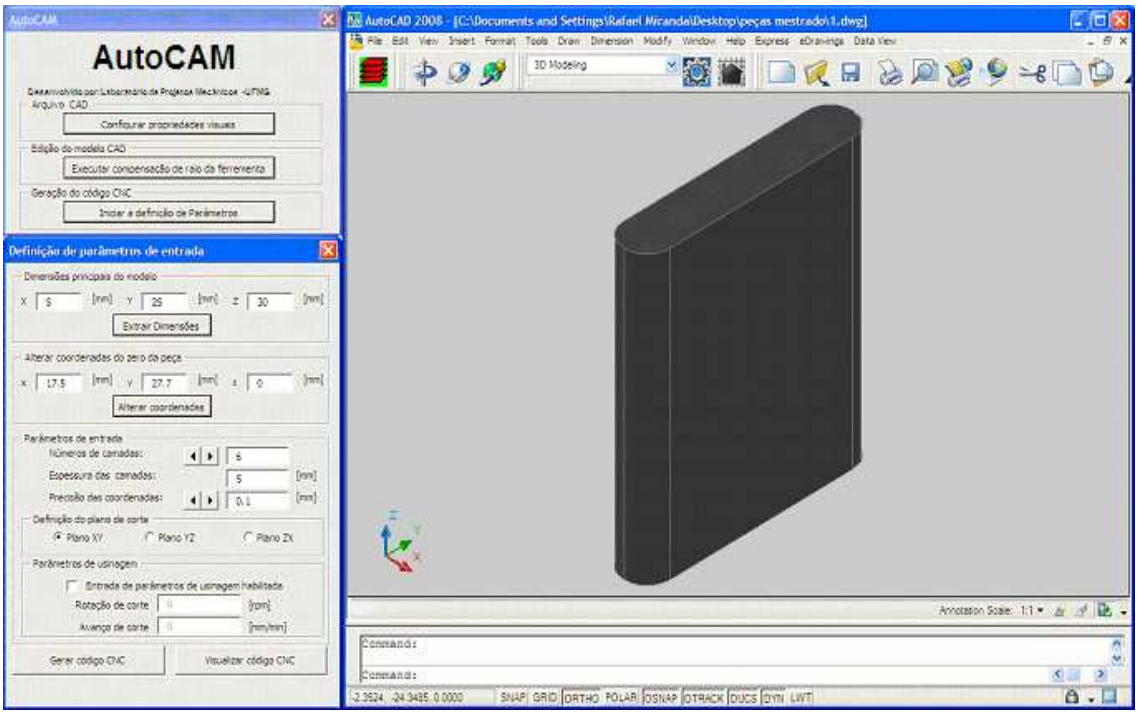


FIG.4.8-Visualização do modelo A (FIG. 4.5) após sua edição, passando a corresponder ao modelo 1 (FIG. 4.3).

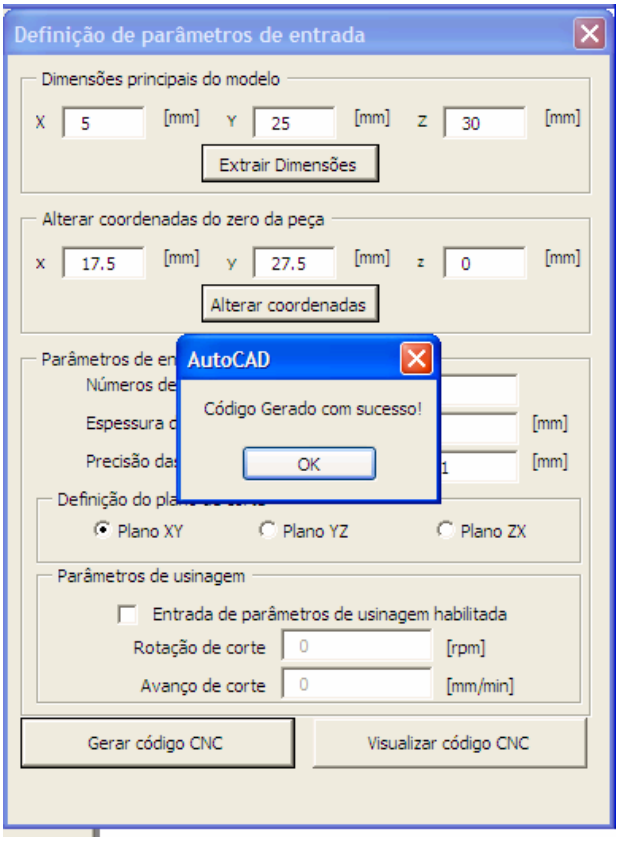


FIG.4.9-Tela de definição de parâmetros: Mensagem de confirmação da geração do código

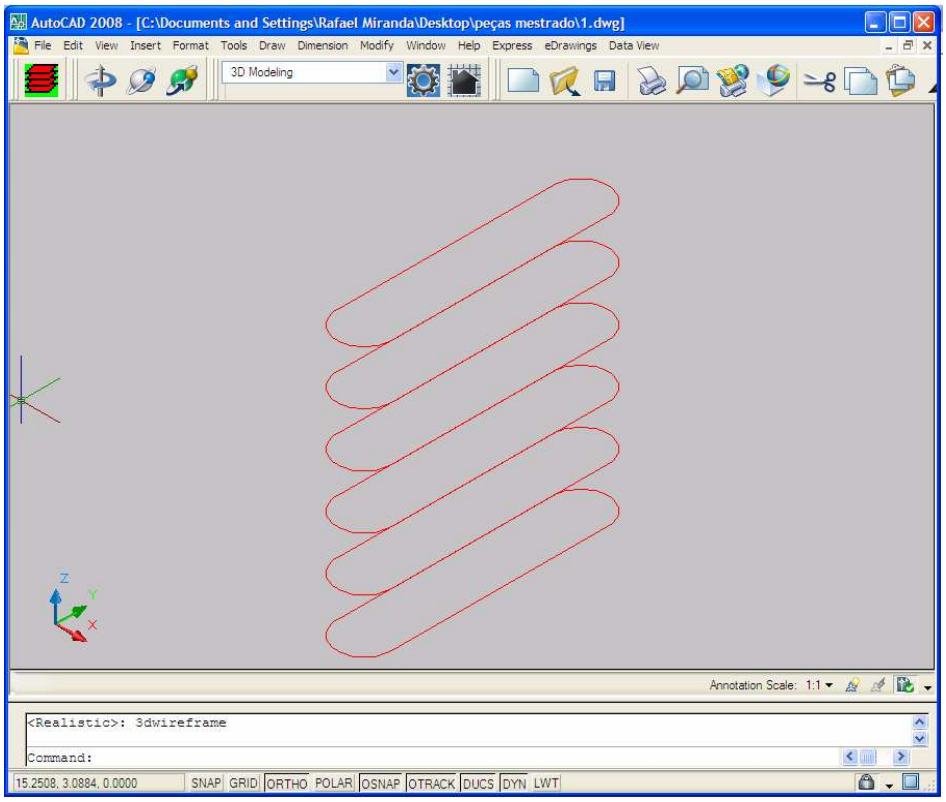


FIG.4.10-Visualização do modelo 1 (FIG. 4.3) após a geração do código CNC.

#INÍCIO DO CÓDIGO#	**G01 X 17.5 Y 30**
	G01 Z-20
G01 X0 Y0	**G01 X 17.5 Y 50**
G01 Z 5	**G02 X 22.5 Y 50 I 2.5 J 0**
G01 X 17.5 Y 30	**G01 X 22.5 Y 30**
G01 Z-5	**G02 X 17.5 I -2.5 J 0**
G01 X 17.5 Y 50	**G01 Z 5**
G02 X 22.5 Y 50 I 2.5 J 0	**G01 Z-20**
G01 X 22.5 Y 30	**G01 X 17.5 Y 30**
G02 X 17.5 Y 30 I -2.5 J 0	**G01 Z-25**
G01 Z 5	**G01 X 17.5 Y 50**
G01 Z-5	**G02 X 22.5 Y 50 I 2.5 J 0**
G01 X 17.5 Y 30	**G01 X 22.5 Y 30**
G01 Z-10	**G02 X 17.5 Y 30 I -2.5 J 0**
G01 X 17.5 Y 50	**G01 Z 5**
G02 X 22.5 Y 50 I 2.5 J 0	**G01 Z-25**
G01 X 22.5 Y 30	**G01 X 17.5 Y 30**
G02 X 17.5 Y 30 I -2.5 J 0	**G01 Z-30**
G01 Z 5	**G03 X 22.5 Y 30 I 2.5 J 0**
G01 Z-10	**G01 X 22.5 Y 50**
G01 X 17.5 Y 30	**G03 X 17.5 Y 50 I -2.5 J 0**
G01 Z-15	**G01 X 17.5 Y 30**
G01 X 17.5 Y 50	**G01 Z 5**
G02 X 22.5 Y 50 I 2.5 J 0	**G01 X 0Y 0**
G01 X 22.5 Y 30	**G01 Z 0**
G02 X 17.5 Y 30 I -2.5 J 0	
G01 Z 5	**#FIM DO CÓDIGO#**
G01 Z-15	

QUADRO 4.1 – Visualização do código CNC gerado para o modelo 1 (FIG. 4.3).

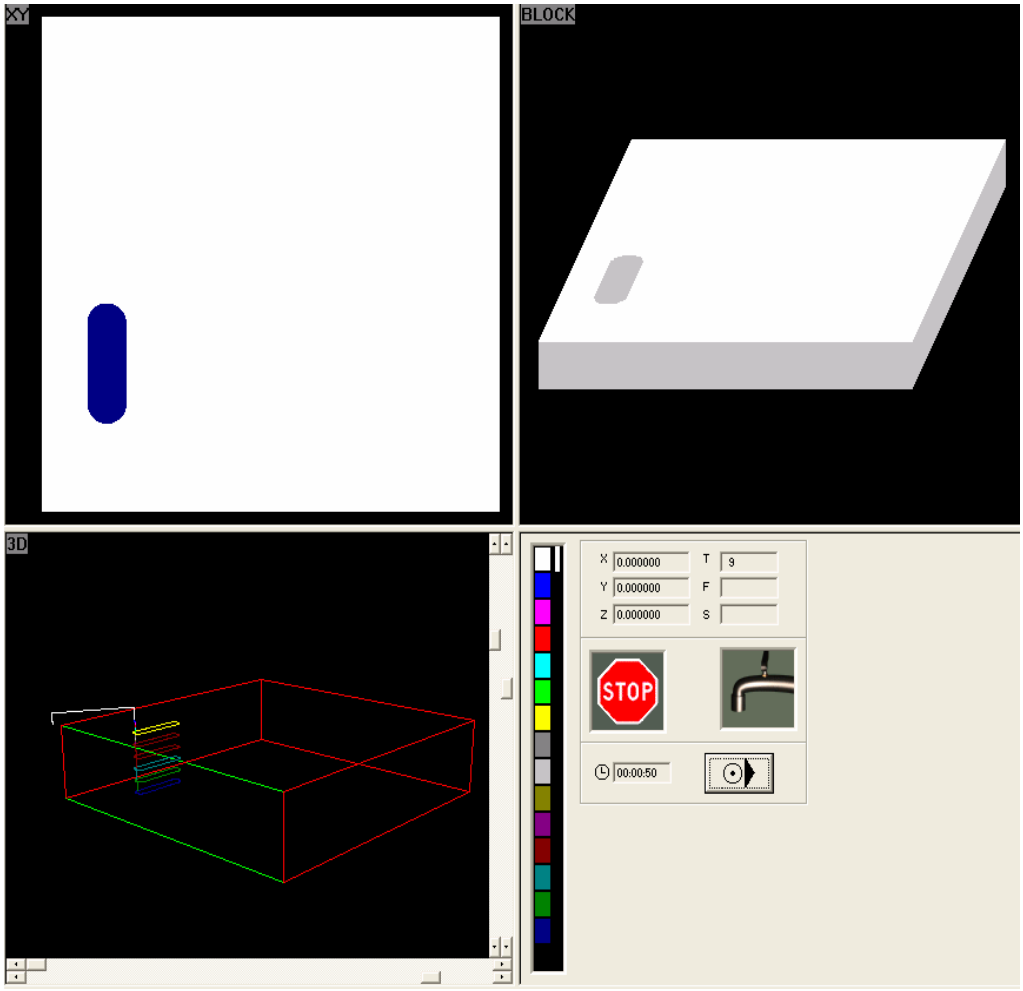


FIG.4.11-Verificação utilizando o *CNC Simulator* da trajetória do código gerado para o modelo 1 (FIG. 4.3).

Para a fabricação do protótipo físico definiu-se a utilização do método de prototipagem por retirada de material, utilizando uma máquina de comando numérico dotada de três eixos (FIG. 4.12), desenvolvida no Laboratório de Projetos Mecânicos, inserido no Programa de Pós-Graduação em Engenharia Mecânica da UFMG (SILVEIRA ,2007).

O *AutoCAM* foi desenvolvido tendo em mente sua aplicação na fabricação de protótipos para fins didáticos. Como envolve então um processo de fabricação, o programa foi desenvolvido de modo a poder ser utilizado aliado aos recursos de fabricação disponíveis.

Já que a máquina disponível possui 3 eixos e o *AutoCAM* possibilita o usuário trabalhar com os três planos ortogonais, houve o cuidado de adequar a estratégia de modo que o eixo de rotação da ferramenta de corte permaneça sempre na vertical.

O modelo foi fabricado utilizando uma peça de isopor (figura 4.13) já preparada com dimensões próximas às desejadas no protótipo.

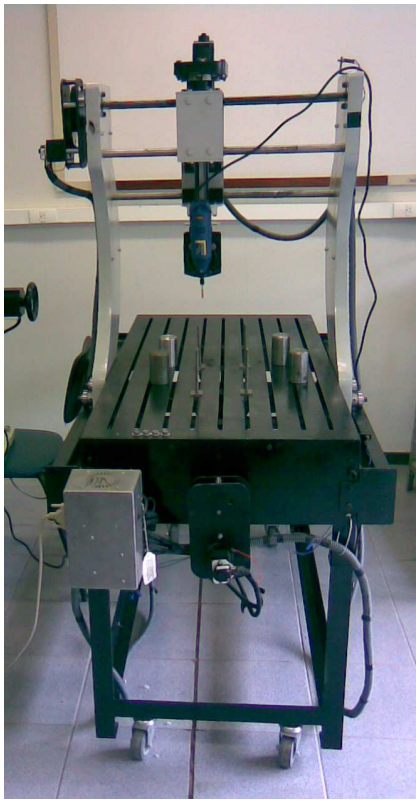


FIG 4.12 – Máquina de comando numérico; dotada de três eixos, utilizada na fabricação do protótipo físico .

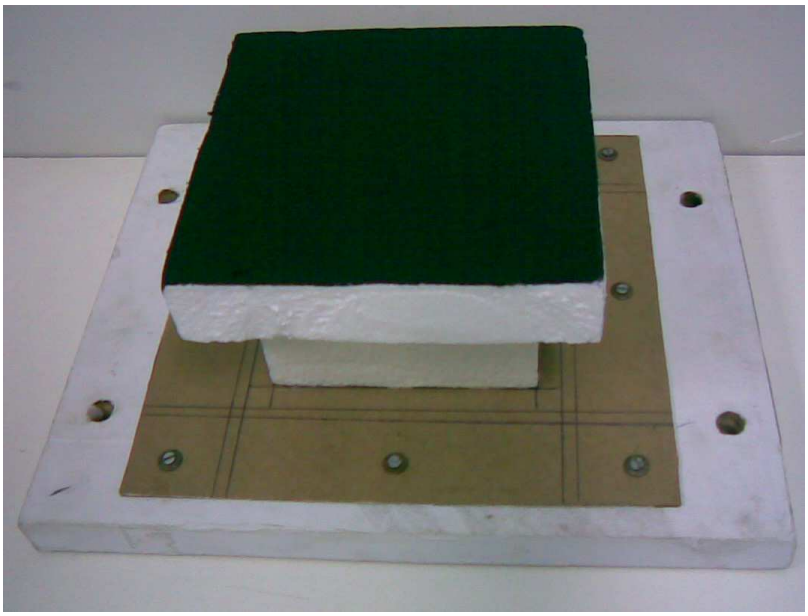


FIG. 4.13 – Peça de isopor utilizada como material base para a fabricação do protótipo físico.

A peça de isopor foi devidamente posicionada na máquina por meio de mecanismos de fixação (FIG. 4.14) apropriados. Estes mecanismos foram devidamente regulados e apertados de modo a conseguir um grau razoável de nivelamento desta peça em relação à máquina CNC.

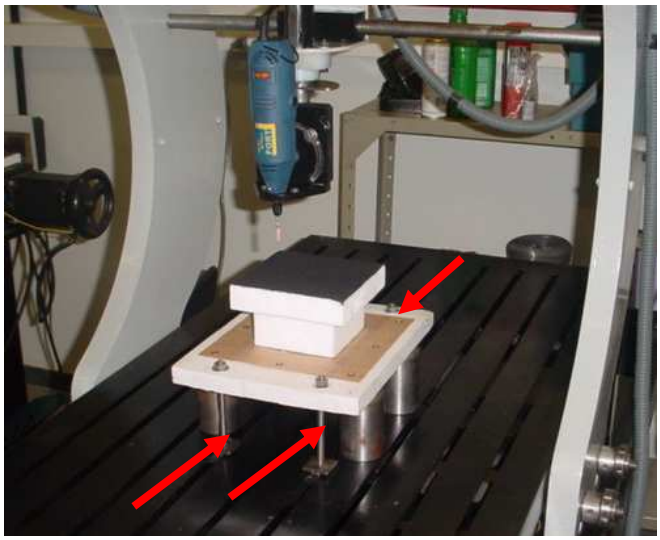


FIG. 4.14 – Fixação da peça de isopor utilizada como material base para a fabricação do protótipo físico.

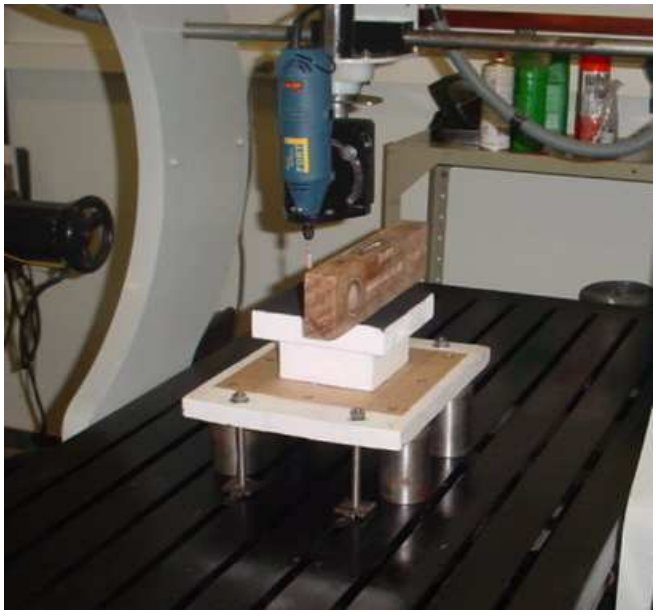


FIG. 4.15 – Nivelamento da peça de isopor utilizada como material base para a fabricação do protótipo físico.

A ferramenta de corte utilizada na usinagem foi um micro-rebolo instalado no cabeçote em uma micro-retífica adaptada à máquina fresadora CNC. Foi utilizado um rebolo

tipo 1B (ASM,1999) cujo diâmetro é de 5mm e sua altura é de 25mm. Abaixo, podem ser vistos na figura 4.16 a micro-retífica e o micro-rebolo utilizado na fabricação do protótipo físico.

FIG. 4.16– Micro-retífica utilizada como ferramenta de corte da máquina de comando numérico.

Para o controle dos motores de passo da máquina CNC de três eixos, foi utilizado *software MaxNC* (FIG. 4.17). Os códigos gerados pelo *AutoCAM* e validados utilizando o *CNCSimulator* foram executados seguindo a seqüência de operações elaborada para a fabricação do protótipo físico. Por conveniência alguns códigos foram editados manualmente e agrupados para agilizar o processo de fabricação.

FIG. 4.17– Interface do *software MaxNC*.

Antes de aplicar a seqüência de códigos numéricos gerados pelo *AutoCAM* foi executado um código elaborado manualmente para fazer furações no modelo físico e prepará-lo para os códigos gerados pelo *AutoCAM.*

Estas furações foram introduzidas de modo a não restar material indesejado nas cavidades introduzidas no modelo físico após a execução dos códigos numéricos.

A figura 18 mostra através de setas vermelhas as furações iniciais realizadas no modelo físico para completar a remoção de material necessária para a obtenção das cavidades desejadas no protótipo físico final. Na mesma figura 18, pode ser vista uma furação no zero estabelecido na peça de isopor, indicado pela seta de cor verde.

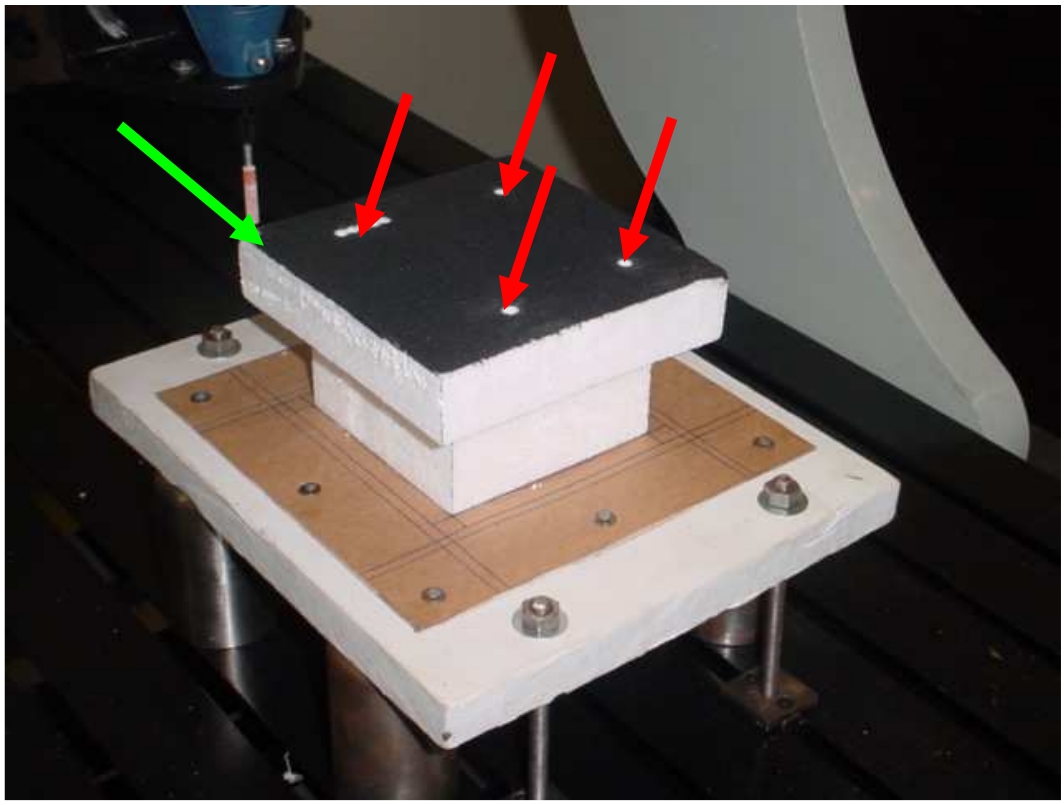


FIG. 4.18– Furações iniciais que precederam à aplicação dos códigos gerados pelo *AutoCAM*.

Adiante na figura 4.19 as setas indicam a execução da cavidade oblonga correspondente aos códigos agrupados dos modelos parciais 1 e 2 da figura 4.3.

A setas vermelhas da figura 4.20 apontam para duas cavidades cilíndricas no protótipo físico, resultantes da execução dos códigos agrupados dos modelos parciais 3 e 4, da figura 4.3. A seta azul aponta para a cavidade cilíndrica correspondente ao agrupamento do código dos modelos parciais 5 e 6, da figura 4.3. Já a seta verde aponta para a cavidade

correspondente ao modelo parcial 9 da figura 4.3. Embora a seqüência elabora previsse a realização da usinagem referente aos modelos parciais 7 e 8 (FIG 4.21), esta inversão aplicada por temor de perder o protótipo devido à baixa resistência do seu material, não afetou em nada o protótipo final.

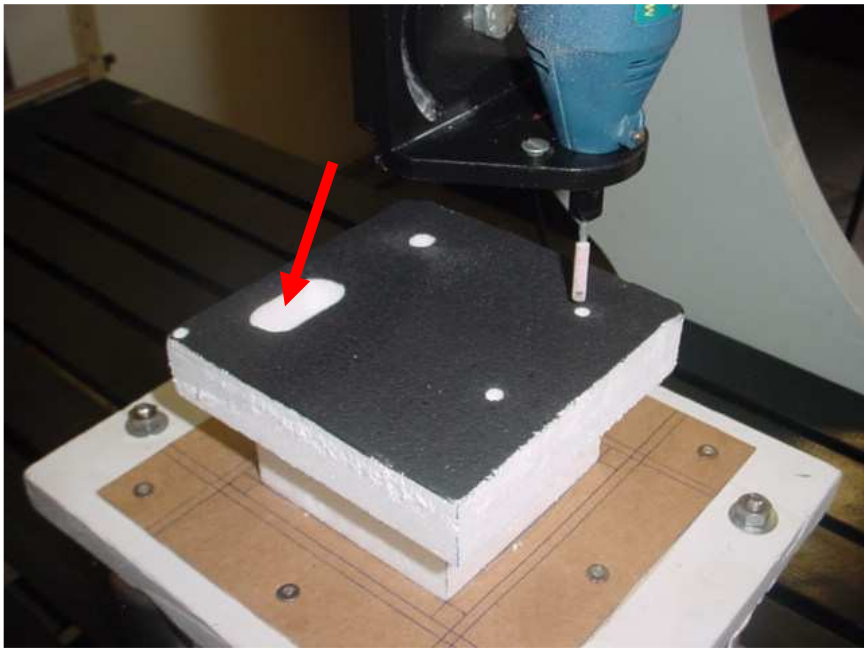


FIG. 4.19–Protótipo após execução dos códigos numéricos dos modelos 1 e 2 FIG. 4.3

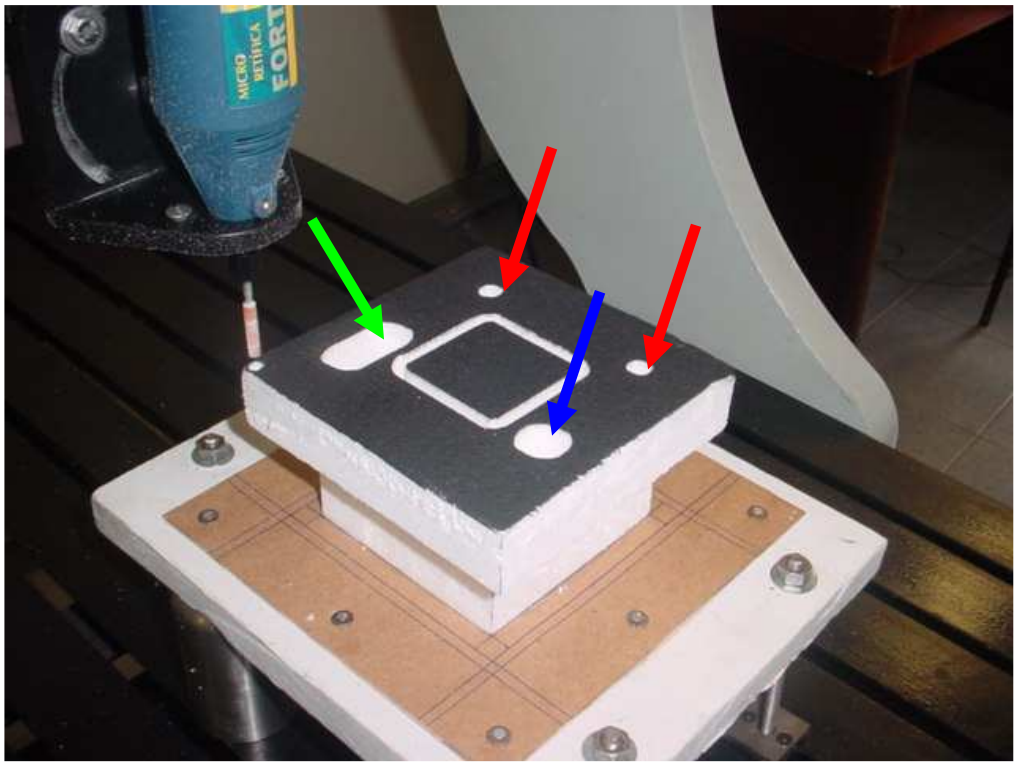


FIG. 4.20–Protótipo após execução dos códigos numéricos dos modelos 3,4 5, 6 e 9 da FIG. 4.3

A inversão da seqüência elaborada também ocorre na execução das cunhas retangular (modelo parcial 12 FIG. 4.3) e triangular (modelo parcial 13 FIG. 4.3) antes, como pode ser visto nas figuras 4.23 e 4.24.

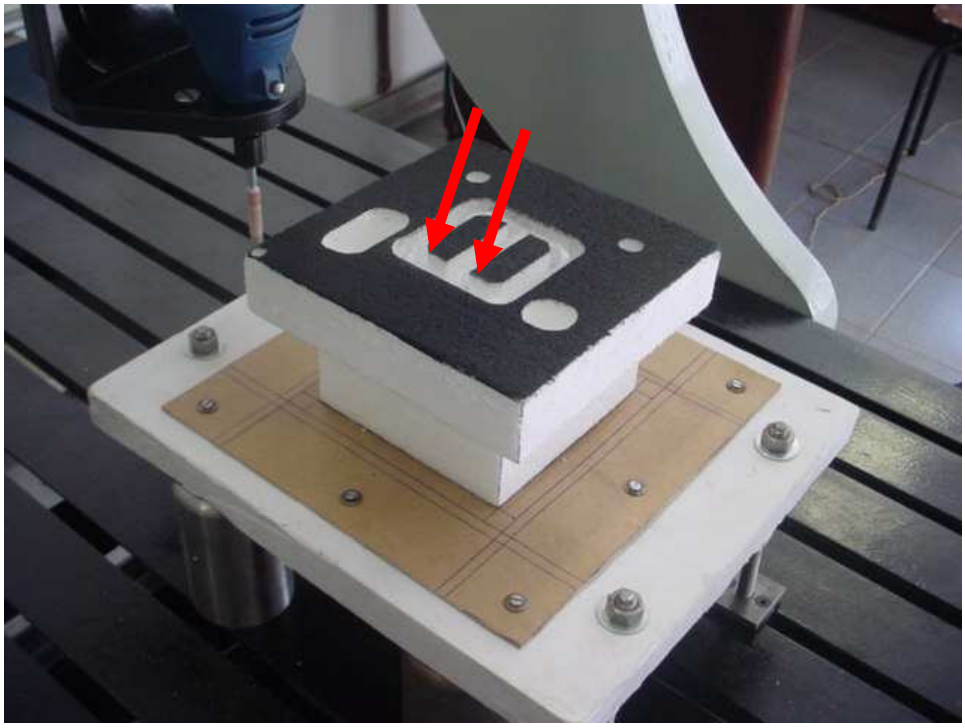


FIG. 4.21–Protótipo após execução dos códigos numéricos dos modelos 7 e 8 da FIG. 4.3

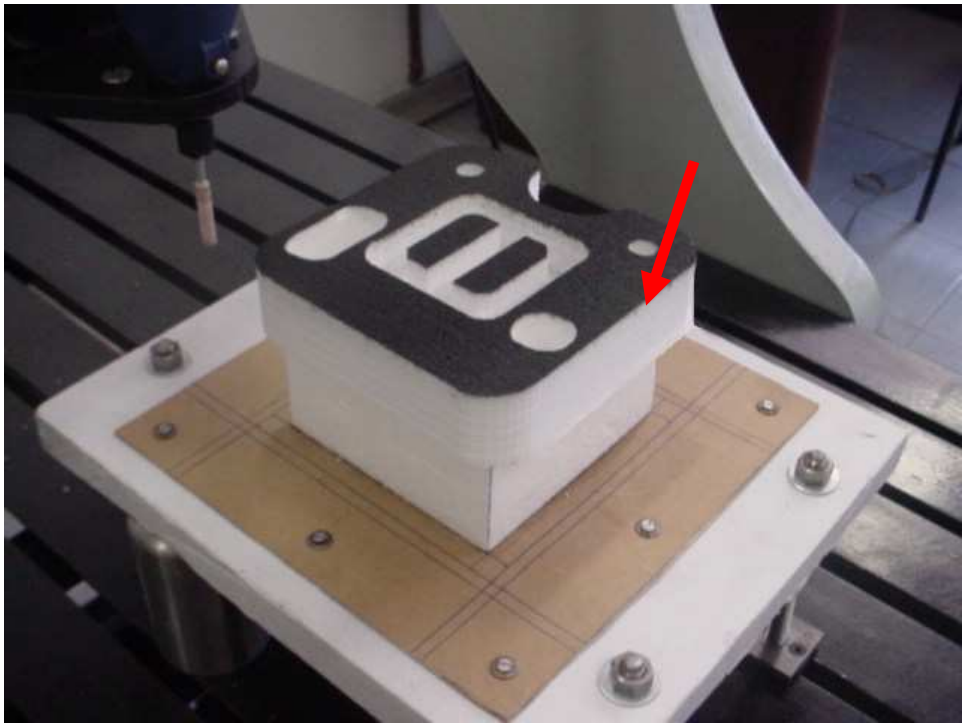


FIG. 4.22–Protótipo após execução do código numérico do modelo 10 da FIG. 4.3

da execução da usinagem referente ao modelo 11 da figura 4.3. Fato que t não acarretou em nenhum prejuízo no protótipo final e vem a salientar a flexibilidade da estratégia elaborada para a utilização do *AutoCAM* na prototipagem por retirada de material .

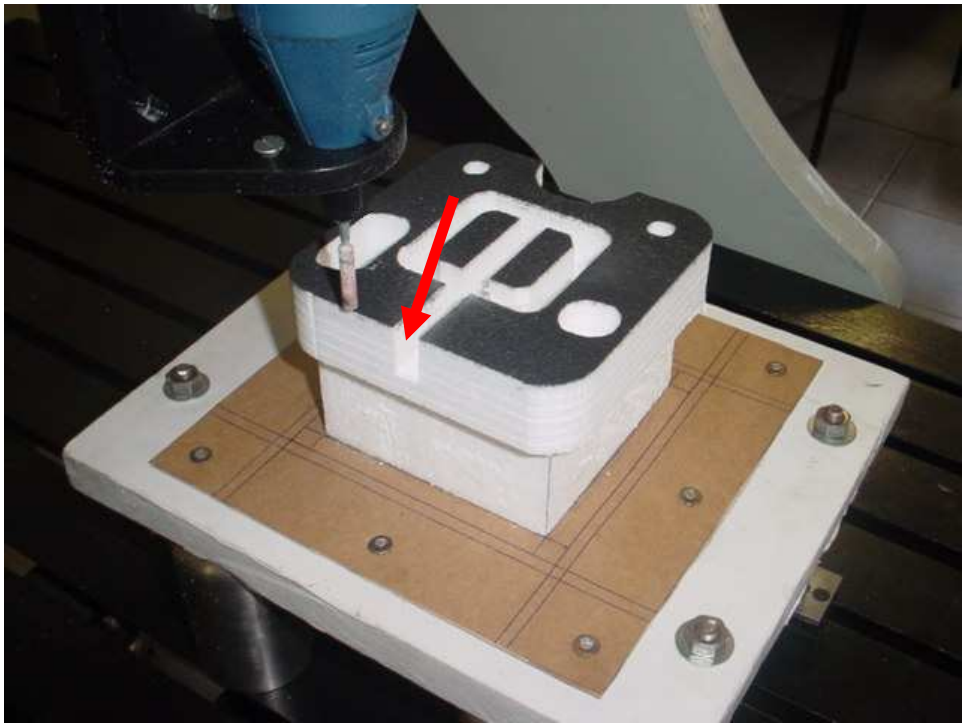


FIG. 4.23–Protótipo após execução do código numérico do modelo 12 da FIG. 4.3

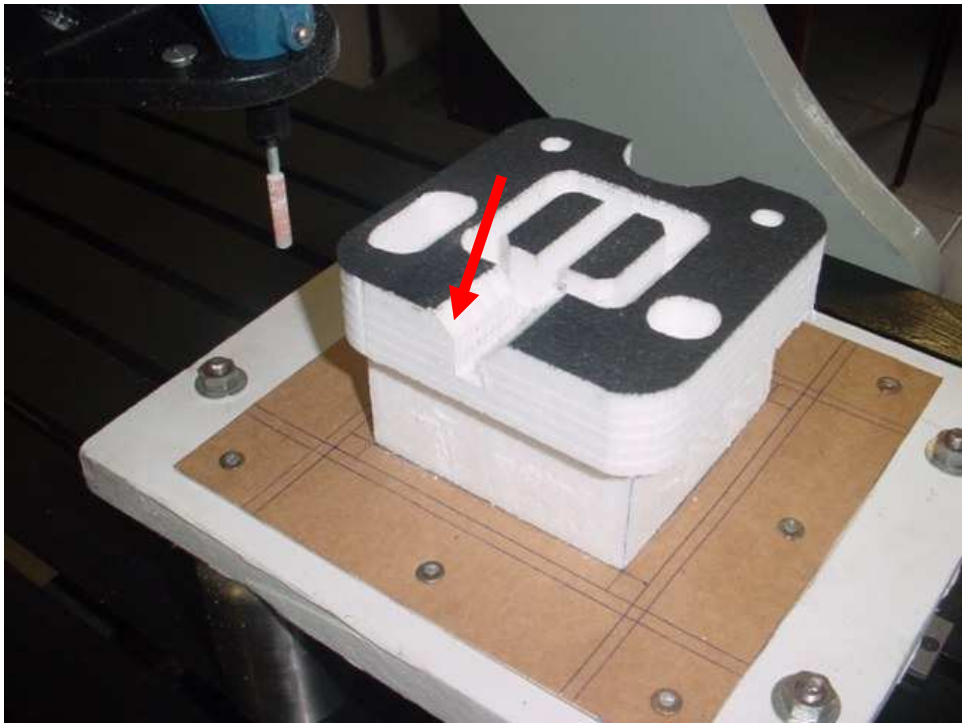


FIG. 4.24–Protótipo após execução do código numérico do modelo 13 da FIG. 4.3

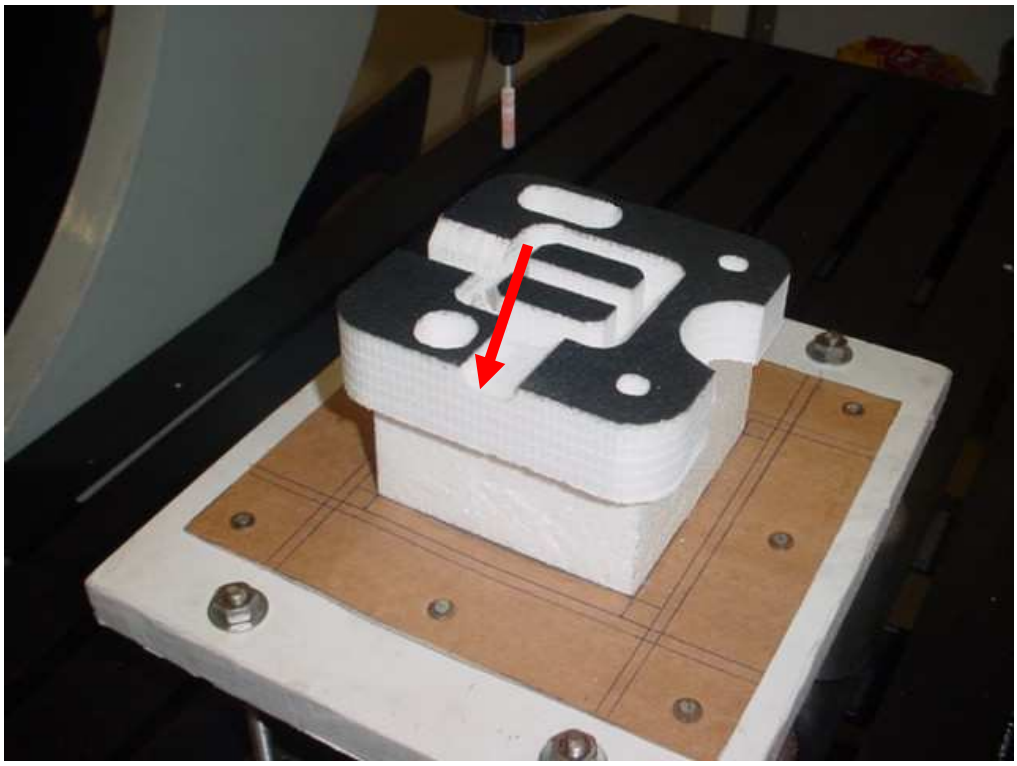


FIG. 4.25–Protótipo após execução do código numérico do modelo 11 da FIG. 4.3

Após a aplicação de todos os códigos gerados foi obtido um protótipo físico final através da aplicação prática do programa *AutoCAM* (FIG. 4.24)

FIG. 4.26–Protótipo físico final obtido através da aplicação prática do programa *AutoCAM* .

5.CONCLUSÕES

Foi demonstrado que o programa computacional *AutoCAM* de fato consegue gerar códigos de comando numérico a partir de dados retirados diretamente de um modelo CAD. Os códigos gerados são passíveis de serem aplicados na fabricação de protótipos físicos utilizando a tecnologia de fabricação por retirada de material. Isto caracteriza neste trabalho a utilização do método do "fatiamento direto" do modelo aplicado à prototipagem por retirada de material, de modo semelhante ao realizado por SANTOS (2002) através do programa computacional *3D FORM 2.0*.

Em comparação dos resultados obtidos no presente trabalho ,que teve como um de seus produtos o programa *AutoCAM*, com os resultados obtidos por SANTOS (2002) em seu estudo, resumido no programa desenvolvido *3D FORM 2.0*., pode-se dizer que existiram avanços nos vieses da estratégia de armazenamento de dados, da estratégia geração de trajetória, do acréscimo de funcionalidades no programa, do desenvolvimento de uma estratégia de utilização do programa e da capacidade de acompanhamento da evolução dos sistemas operacionais e CAD.

A estratégia de ordenação dos dados referentes aos objetos 2D gerados no ato do fatiamento do objeto 3D em camadas, utilizada no *AutoCAM* adota a ordenação dos objetos a cada camada e não a ordenação de todas as camadas simultaneamente, como o realizado por SANTOS (2002). O reflexo desta ação é o ganho de um nível de ordenação que não precisa ser realizado (que não precisa ser realizado).

O armazenamento dos dados extraídos dos objetos 2D no *AutoCAM,* foi realizado através da alocação dinâmica de memória e não estática (com número fixo de elementos). Este fato acarretou na diminuição da quantidade de arquivos necessários para todo o processo de extração e ordenação até a obtenção do código numérico de três arquivos para somente um, o que contém o código numérico. Também como conseqüência da aplicação da alocação dinâmica de memória, tem-se o fato de que o custo computacional foi otimizado pois, a alocação estática implica no super dimensionamento das matrizes no caso de uma camada possuir poucos objetos 2D , e; no caso de existirem muitos objetos, existe o risco de se extrapolar a capacidade de armazenamento.

Ainda com relação à estratégia de geração de trajetória, é válido frisar que esta foi desenvolvida com os objetivos de evitar a colisão indesejada da ferramenta com a peça

durante sua translação e redução deste tempo dentro do qual a ferramenta desloca mas não executa o corte.

A respeito do avanço no acréscimo de funcionalidades, isto é traduzido no recurso de compensação de raio da ferramenta oferecido pelo *AutoCAM*, realizado por meio da edição do modelo 3D.

Em se tratando do acompanhamento da evolução tanto dos sistemas operacionais quanto dos sistemas CAD disponíveis, é fato que o *AutoCAM* tem plenas condições de acompanhar a evolução do *AutoCAD*® e inclusive pode ser utilizado, com possível necessidade de adaptação, em versões anteriores do *AutoCAD*®, bem como é capaz de acompanhar a evolução do *Windows*®.

Em contrapartida, não foi obtido êxito na proposta inicial de proceder como o realizado por SANTOS (2002) na utilização de um *software* visualizador embutido por assim dizer, em seu programa computacional nem na obtenção do programa sob a forma de executável, devido a limitações da plataforma de programação do*AutoCAD VBA*. De forma alguma isso desprestigia o *AutoCAM*, pois estas características embora desejáveis poderiam inviabilizar a continuidade do desenvolvimento do programa uma vez que pode não existir um visualizador mais atual que o utilizado por SANTOS (2002) que seja manipulável segundo a tecnologia *ActiveX* de automação e, a capacidade do *AutoCAM* de se realizar a compensação de raio da ferramenta poderia ser ameaçada.

Também como produto deste trabalho tem-se a estratégia de utilização do programa *AutoCAM*, que requer do usuário utilizar quantos modelos 3D forem necessários para representar as geometrias da peça considerando-se os contornos externos e as cavidades, de tal forma que a somatória destas ações se traduza no protótipo final. Esta estratégia não só vem a servir para a aplicação do *AutoCAM,* mas também da compreensão e aplicação do processo de prototipagem por retirada de material.

A validação de toda a tecnologia assimilada ou desenvolvida, agregada ao *AutoCAM* foi expressa de modo virtual através a utilização do *software CNCsimulator* e de modo físico em um protótipo, obviamente físico, que vem também a constituir um produto do presente trabalho.

O objetivo geral de contribuir para o domínio da tecnologia de fabricação de protótipos produzindo peças destinadas ao auxílio do ensino aos estudantes de Engenharia Mecânica em disciplinas relacionadas às áreas de processos de fabricação, desenho mecânico e projeto de máquinas; e os objetivos específicos do exercício de conhecimentos de controle de programas CAD através do uso de interfaces gráficas e o desenvolvimento de estratégias

para fabricação de peças utilizando a técnica similar a de prototipagem rápida; foram atingidos.

Futuramente objetiva-se o uso do método de fabricação de protótipos por adição de material, uma tecnologia mais recente do que a retirada de material, e que oferece um campo propício para pesquisas, pois embora já sejam comercializados equipamentos de prototipagem rápida para uso industrial, são poucos os equipamentos encontrados para fins didáticos e de baixo custo.

ABSTRACT

In this work is presented a didactic software developed that promotes an automation for code applied in CNC machines (the G code) generation process, this is done by extracting data from 3D models created in some CAD platform. This software developed using the Visual Basic programming language is able to control the CAD system, AutoCAD® release 2008, externally by using an ActiveX type automation technology known as VBA (Visual Basic for Applications). A 3D model and some user's parameters are the entries for this software and a CNC code directly applicable in CNC machines is generated as output. This code has an embedded strategy similar to that used by the Rapid Prototyping method to build a part. In this case, this part is built considering all the layers generated from the 3D model's direct slicing modeled in some CAD system. To validate the developed software; a free manufacturing process simulation software, named CNCSimulator and physical prototype manufactured, by a CNC machine, using prototyping by material removal, were used. The developed software, the application and the used strategy to manufacture the designed part are considered the products of the present work. The work validates the proposal to use the program as a tool in teaching subjects related to the mechanical design and manufacturing processes within the scope in the field of Engineering. The use of the AutoCAM software requires from the user the development of strategies that allow representing the geometry of a part using how many 3D models as needed to represent the external contours and cavities, such that the sum of these actions will result in the final prototype.

Keywords: *CAD, CAM, CNC, direct slicing, G code*, *Rapid Prototyping, Didactic software.*

REFERÊNCIAS BIBLIOGRÁFICAS

1. ASM International. ASM Handbook- Vol.16-Machining.3.ed.USA: ASM International, 1999.p.1-1089

2. AutoDesk, *ActiveX and VBA Developers' Guide 2000*, 1 ed.USA: Autodesk, Inc., Mar.1999, p. 1-432.

3. AutoDesk, *ActiveX and VBA Developers' Guide 2004*, 1 ed.USA: Autodesk, Inc., Feb.2003, p. 1-368.

4. AutoDesk, *AutoCAD 2007 Users' Guide*, 1 ed.USA: Autodesk, Inc., Feb.2006, p. 1-1236.

5. AutoDesk, *Site oficial da Autodesk, Inc.*, Disponível em: <http://www.autodesk.com.br> Acesso em 20/01/2009.

6. Bellini,A.; Güçeri,S., *Mechanical characterization of parts fabricated using fused deposition modeling.* Rapid Prototyping Journal, United Kingdom, v.9, n.4, p.252-264, 2003.

7. Chang,C.C., *Rapid prototyping fabricated by UV resin spray nozzles*, . Rapid Prototyping Journal, United Kingdom, v.10, n.2, p.136-145, 2004.

8. Chakraborty, D; Choudhury, A. R., *A semi-analytic approach for direct slicing of free form surfaces for layered manufacturing.* Rapid Prototyping Journal, United Kingdom, v.13, n.4, p.256-264, 2007.

9. Chiu, Y.Y.; Liao, Y.S., *Laser path planning of burn-out rule for LOM process.* Rapid Prototyping Journal, United Kingdom, v.9, n.4, p.201-211, 2003.

10. Choi;S.H; Kwok, K.T., *A topological hierarchy-sortingalgorithm for layered manufacturing.* Rapid Prototyping Journal, United Kingdom, v.10, n.2, p.98-113, 2004.

11. De Beer, D. J.;Barnard,L. J.; Booysen, G. J,*Three-dimensional plotting as a visualisation aid for architectural use.* Rapid Prototyping Journal, United Kingdom, v.10, n.2, p.146-151, 2004.

12. Dolenc, A.; Mäkelä, I., *Rapid prototyping from a computer scientist's point of view.* Rapid Prototyping Journal, United Kingdom, v.2, n.2, p.18-25, 1996.

13. Fadel, G. M.;Kirschman, C., *Accuracy issues in CAD to RP translations.* Rapid Prototyping Journal, United Kingdom, v.2, n.2, p.41-48, 1996.

14. Falck, D., *EMC Handbook- G-Code Programming Basics*. Disponível em: < http://www.linuxcnc.org/handbook/gcode/g-code.html> Acesso em 02/12/2008.

15. Finkelstein, E., *AutoCAD 2008 and AutoCAD 2008 LT Bible*, 1 ed. USA: Wiley Publishing, Inc., 2007, p.1-1124.

16. Finkelstein, E., *AutoCAD 2005 and AutoCAD 2005 LT Bible*, 1 ed. USA: Wiley Publishing, Inc., 2004, p.1-1251.

17. Gibson,I.; Kvan,T.; Ming,L. W., *Rapid prototyping for architectural models*. Rapid Prototyping Journal, United Kingdom, v.8, n.2, p.91-99, 2002.

18. Gopakumar, S,*RP in medicine: a case study in cranial reconstructive surgery.* Rapid Prototyping Journal, United Kingdom, v.10, n.3, p.207-211, 2004.

19. Greenwald, I. D.; Maureen K., *The Share 709 System: Programming and Modification* Journal of the ACM, New York, NY, USA: ACM, p.128–133, 1959.

20. Guangchun, W.;Huiping L.;Yanjin G;Guoqun Z. A *rapid design and manufacturing system for product development application*. Rapid Prototyping Journal, United Kingdom, v.10, n.3, p.200-206, 2004.

21. Jamieson, R.; Hacker, H., *Direct slicing of CAD models for rapid prototyping*. Rapid Prototyping Journal, United Kingdom, v.1, n.5, p.4-12, 1995.

22. Jurrens, K. K., *Standards for the rapid prototyping industry*. Rapid Prototyping Journal, United Kingdom, v.5, n.4, p.169-178, 1999.

23. Kumar, C.;Choudhury, R., *Volume deviation in direct slicing.* Rapid Prototyping Journal, United Kingdom, v.11, n.3, p.174-184, 2005.

24. Microsoft, *Microsoft Developer Network - Biblioteca MSDN,*2009, Disponível em: < http://msdn.microsoft.com/pt-br/library/default.aspx>. Acesso em 12/01/2009.

25. Morgan,R.;Sutcliffe, C.J.; O'Neill, W., *Experimental investigation of nanosecond pulsed Nd:YAG laser re-melted pre-placed powder beds.* Rapid Prototyping Journal, United Kingdom, v.7, n.3, p.159-172, 2001.

26. Park,J.;. Tari, M. J.; Hahn, H. T.,*Characterization of the laminated object manufacturing (LOM) process.* Rapid Prototyping Journal, United Kingdom, v.6, n.1, p.36-49, 2000.

27. Perry, G., *Sams Teach Yourself Visual Basic 6 in 21 Days, Professional Reference Edition.* 1. ed USA: Macmillan Computer Publishing, 1999, p.1-1155.

28. Pertence, A. E. M. ; Santos, D. M. C. ; Jardim, H. V. . *Desenvolvimento de Modelos Didáticos para o Ensino de Desenho Mecânico Utilizando o Conceito de Prototipagem Rápida*. In: XXIX Congresso Brasileiro de Ensino de Engenharia Mecânica, 2001,
29. Gramado/ Rio Grande do Sul. Anais do XXIX Congresso Brasileiro de Ensino de Engenharia Mecânica, 2001.

30. Qiu, D.; Langrana, N.A.; Danforth, S.C.; Safari, A.; Jafari, M., *Intelligent toolpath for extrusiom-bassed LM.* Rapid Prototyping Journal, United Kingdom, v.7, n.1, p.18-23, 2001.

31. Sanghera,B.;Naique, S.; Papaharilaou, Y., *Preliminary study of rapid prototype medical models.* Rapid Prototyping Journal, United Kingdom, v.7, n.5, p.275-284, 2001.

32. Santos, D. M. C. *Desenvolvimento de um Programa Computacional para a Prototipagem Rápida por Retirada de Material*. Dissertação (Mestrado em Engenharia Mecânica) – Laboratório de Projetos Mecânicos, Programa de Pós-Graduação em Engenharia Mecânica. Belo Horizonte: Universidade Federal de Minas Gerais, 2002.

33. Silveira, R. C. A., *Desenvolvimento de um Equipamento Mecânico com Controle Numérico Computadorizado para Produção de Protótipos em Escala*. 2007. Dissertação (Mestrado em Engenharia Mecânica) - Laboratório de Projetos Mecânicos, Programa de Pós-Graduação em Engenharia Mecânic . Belo Horizonte: Universidade Federal de Minas Gerais, 2007.

34. Tata, K; Fadel G.; Bagchi, A.; Aziz, N*, Efficient slicing for layered manufacturing*. Rapid Prototyping Journal, United Kingdom, v.4, n.4, p.151-167, 1998.

35. Tong,K.; Lehtihet, E. A; Joshi S., *Parametric error modeling and software error compensation for rapid prototyping.* Rapid Prototyping Journal, United Kingdom, v.9, n.5, p.301-313, 2003.

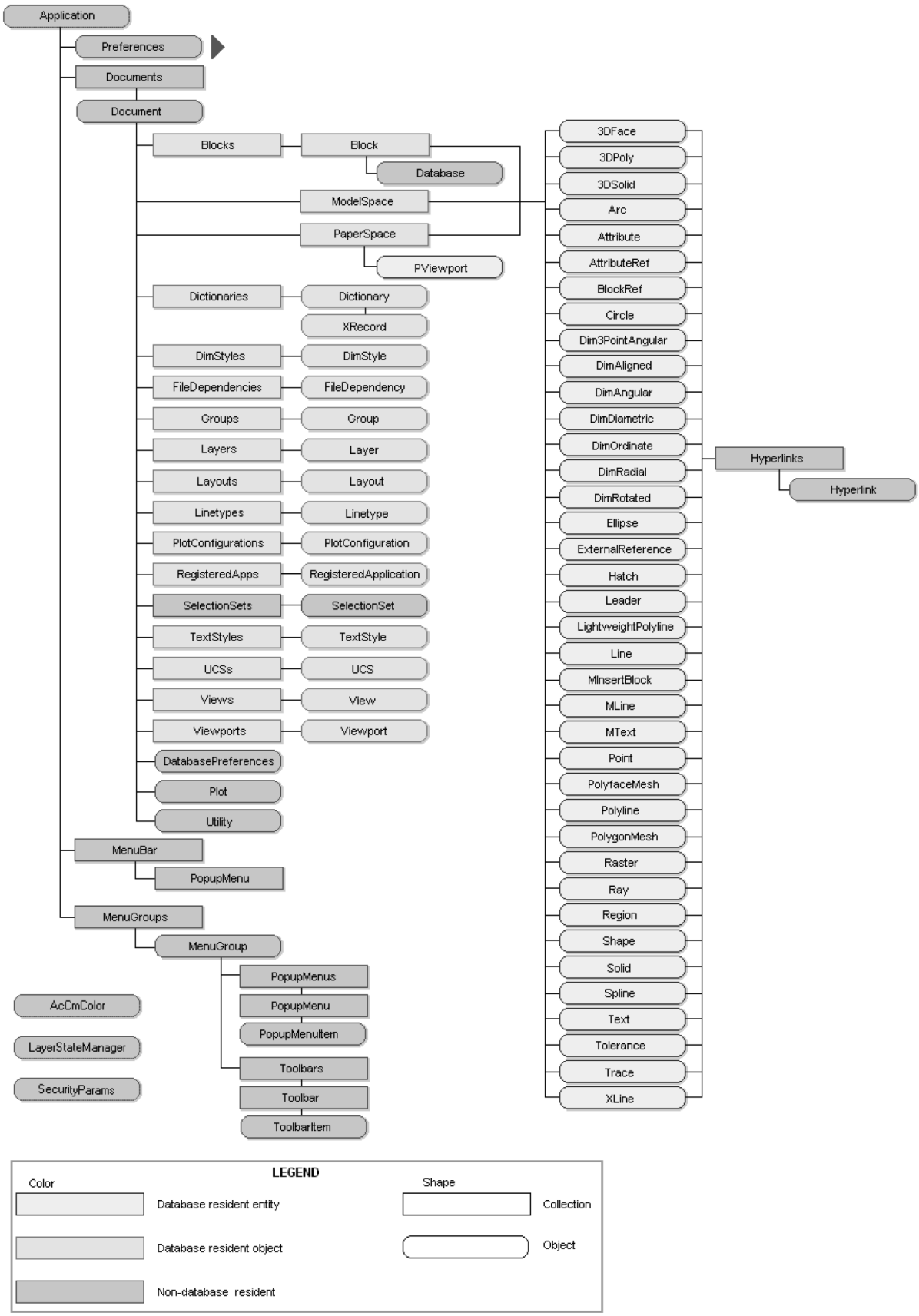
36. Volpato, N., *Prototipagem Rápida-Tecnologia e Aplicações.* Primeira Edição, São Paulo: Edgard Blücher, 2007

37. Walsh, R. A.; Cormier, D. R. *McGraw-Hill machining and metalworking handbook*, 3 ed. USA: McGraw-Hill Companies, Inc, 2006, p 1-974.

38. Yang, Y.;Loh, H.T.;Fuh, J.Y.H.;Wang, Y.G., *Equidistant path generation for improving scanning efficiency in layered manufacturing.* Rapid Prototyping Journal, United Kingdom, v.8, n.1, p.30-37, 2002.

ANEXOS

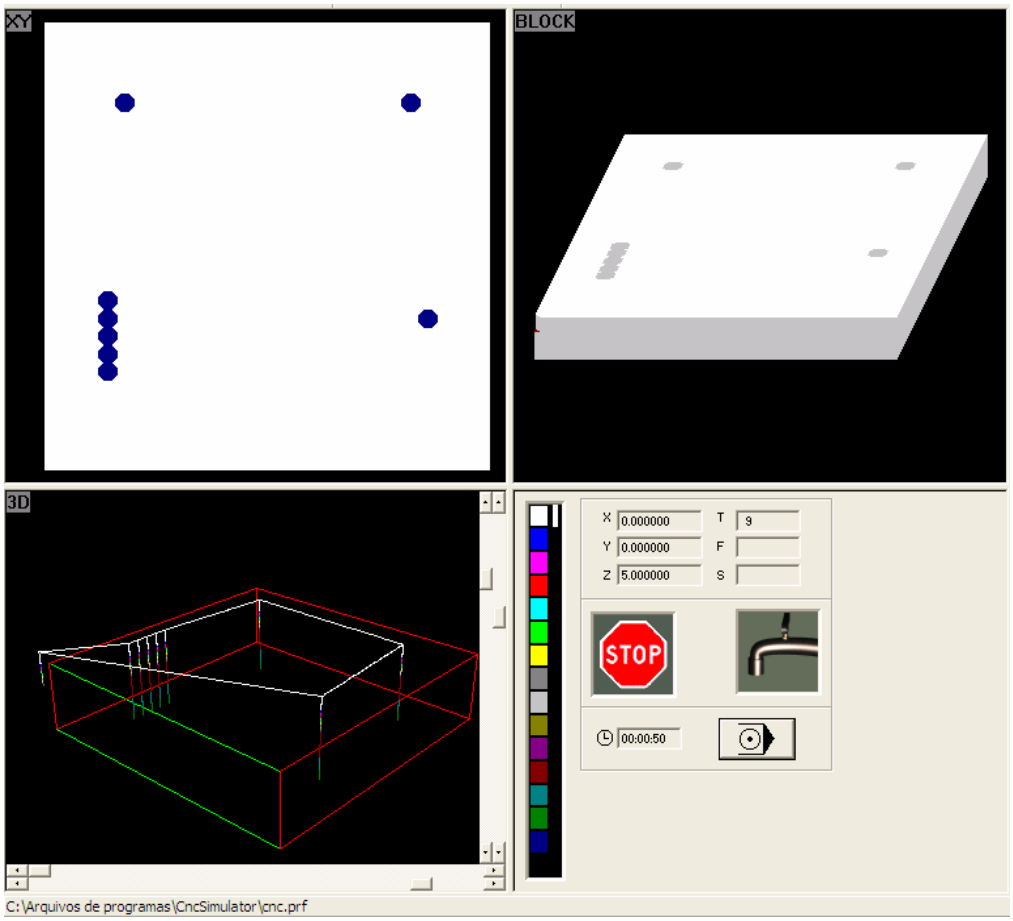
ANEXO I- RELAÇÃO DOS OBJETOS COMPREENDIDOS DENTRO DA TECNOLOGIA ACTIVE X



ANEXO II- UTILIZAÇÃO DO *AUTOCAM* NA GERAÇÃO DOS CÓDIGOS DOS MODELOS PARCIAIS

Código numérico da furação inicial

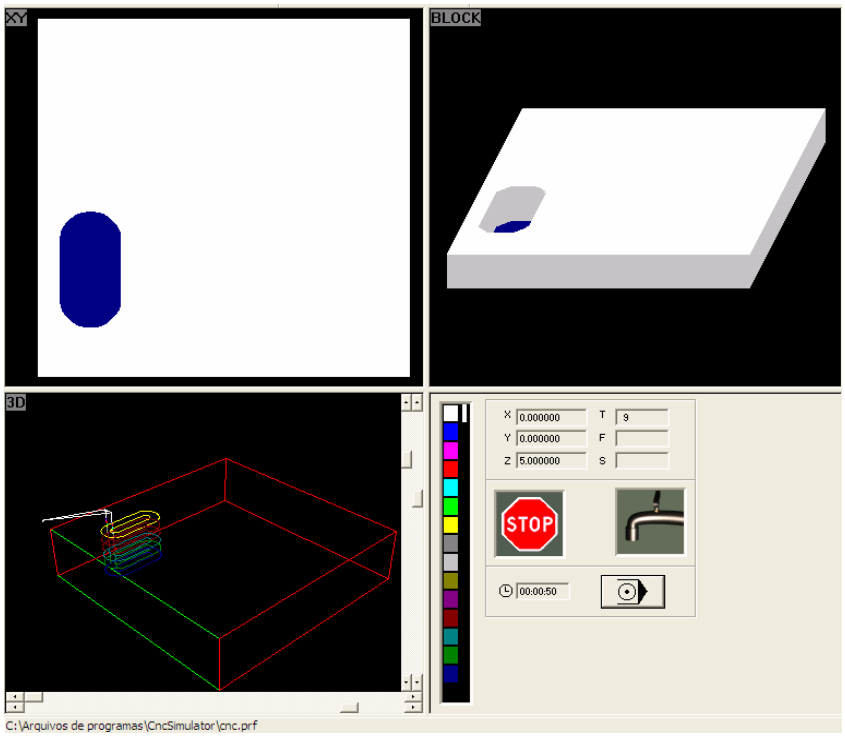
```
G01 X 0 Y 0 Z 5
G01 Z -10
G01 Z 5
G01 X 20 Y 30
G01 Z -10
G01 Z 5
G01 Z -20
G01 Z 5
G01 Z -30
G01 Z 5
G01 X 20 Y 35
G01 Z -10
G01 Z 5
G01 Z -20
G01 Z 5
G01 Z -30
G01 Z 5
G01 X 20 Y 40
G01 Z -10
G01 Z 5
G01 Z -20
G01 Z 5
G01 Z -30
G01 Z 5
G01 X 20 Y 45
G01 Z -10
G01 Z 5
G01 Z -20
G01 Z 5
G01 Z -30
G01 Z 5
G01 X 20 Y 50
G01 Z -10
G01 Z 5
G01 Z -20
G01 Z 5
G01 Z -30
G01 Z 5
G01 X 25 Y 105
G01 Z -10
G01 Z 5
G01 Z -20
G01 Z 5
G01 Z -30
G01 Z 5
G01 X 105 Y 105
G01 Z -10
G01 Z 5
G01 Z -20
G01 Z 5
G01 Z -30
G01 Z 5
G01 X 110 Y 45
G01 Z -10
G01 Z 5
G01 Z -20
G01 Z 5
G01 Z -30
G01 Z 5
GO1 X0 Y0 Z5
```



Visualização do código numérico da furação inicial

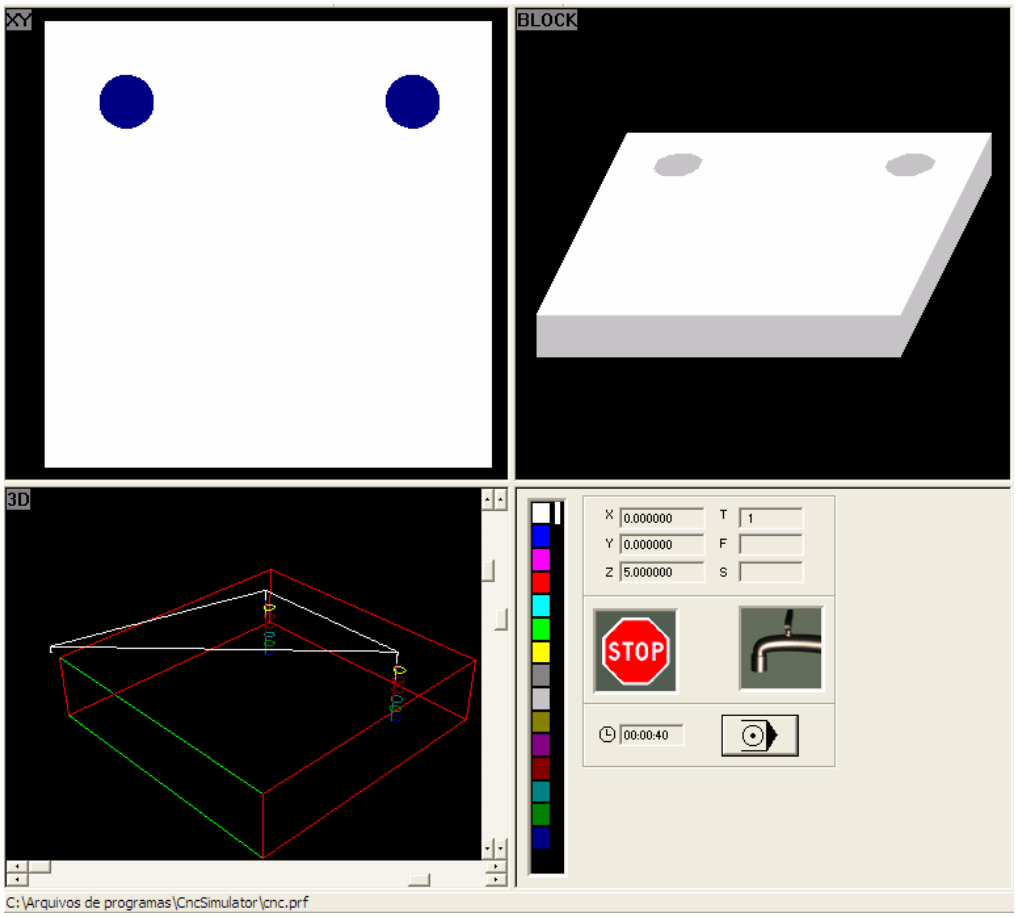
Código numérico agrupado dos modelos parciais 1 e 2.

```
G01 X0 Y0
G01 Z5
G01 X 17.5 Y 30
G01 Z-5
G01 X 17.5  Y 50
G02 X 22.5  Y 50 I 2.5 J 0
G01 X 22.5  Y 30
G02 X 17.5  Y 30 I-2.5 J 0
G01 Z 5
G01 X 17.5 Y 30
G01 Z-10
G01 X 17.5  Y 50
G02 X 22.5  Y 50 I 2.5 J 0
G01 X 22.5  Y 30
G02 X 17.5  Y 30 I-2.5 J 0
G01 Z 5
G01 X 17.5 Y 30
G01 Z-15
G01 X 17.5  Y 50
G02 X 22.5  Y 50 I 2.5 J 0
G01 X 22.5  Y 30
G02 X 17.5  Y 30 I-2.5 J 0
G01 Z 5
G01 X 17.5 Y 30
G01 Z-20
G01 X 17.5  Y 50
G02 X 22.5  Y 50 I 2.5 J 0
G01 X 22.5  Y 30
G02 X 17.5  Y 30 I-2.5 J 0
G01 Z 5
G01 X 17.5 Y 30
G01 Z-25
G01 X 17.5  Y 50
G02 X 22.5  Y 50 I 2.5 J 0
G01 X 22.5  Y 30
G02 X 17.5  Y 30 I-2.5 J 0
G01 Z 5
G01 X 17.5 Y 30
G01 Z-30
G03 X 22.5  Y 30 I 2.5 J 0
G01 X 22.5  Y 50
G03 X 17.5  Y 50 I -2.5 J 0
G01 X 17.5  Y 30
G01 Z 5
G01 X 12.5 Y 30
G01 Z-5
G01 X 12.5  Y 50
G02 X 27.5  Y 50 I 7.5 J 0
G01 X 27.5  Y 30
G02 X 12.5  Y 30 I -7.5 J 0
G01 Z 5
G01 X 12.5 Y 30
G01 Z-10
G01 X 12.5  Y 50
G02 X 27.5  Y 50 I 7.5 J 0
G01 X 27.5  Y 30
G02 X 12.5  Y 30 I -7.5 J 0
G01 Z 5
G01 X 12.5 Y 30
G01 Z-15
G01 X 12.5  Y 50
G02 X 27.5  Y 50 I 7.5 J 0
G01 X 27.5  Y 30
G02 X 12.5  Y 30 I -7.5 J 0
G01 Z 5
G01 X 12.5 Y 30
G01 Z-20
G01 X 12.5  Y 50
G02 X 27.5  Y 50 I 7.5 J 0
G01 X 27.5  Y 30
G02 X 12.5  Y 30 I -7.5 J 0
G01 Z 5
G01 X 12.5 Y 30
G01 Z-25
G01 X 12.5  Y 50
G02 X 27.5  Y 50 I 7.5 J 0
G01 X 27.5  Y 30
G02 X 12.5  Y 30 I -7.5 J 0
G01 Z 5
G01 X 12.5 Y 30
G01 Z-30
G03 X 27.5  Y 30 I 7.5 J 0
G01 X 27.5  Y 50
G03 X 12.5  Y 50 I -7.5 J 0
G01 X 12.5  Y 30
G01 Z 5
G01 X0 Y0 Z5
```



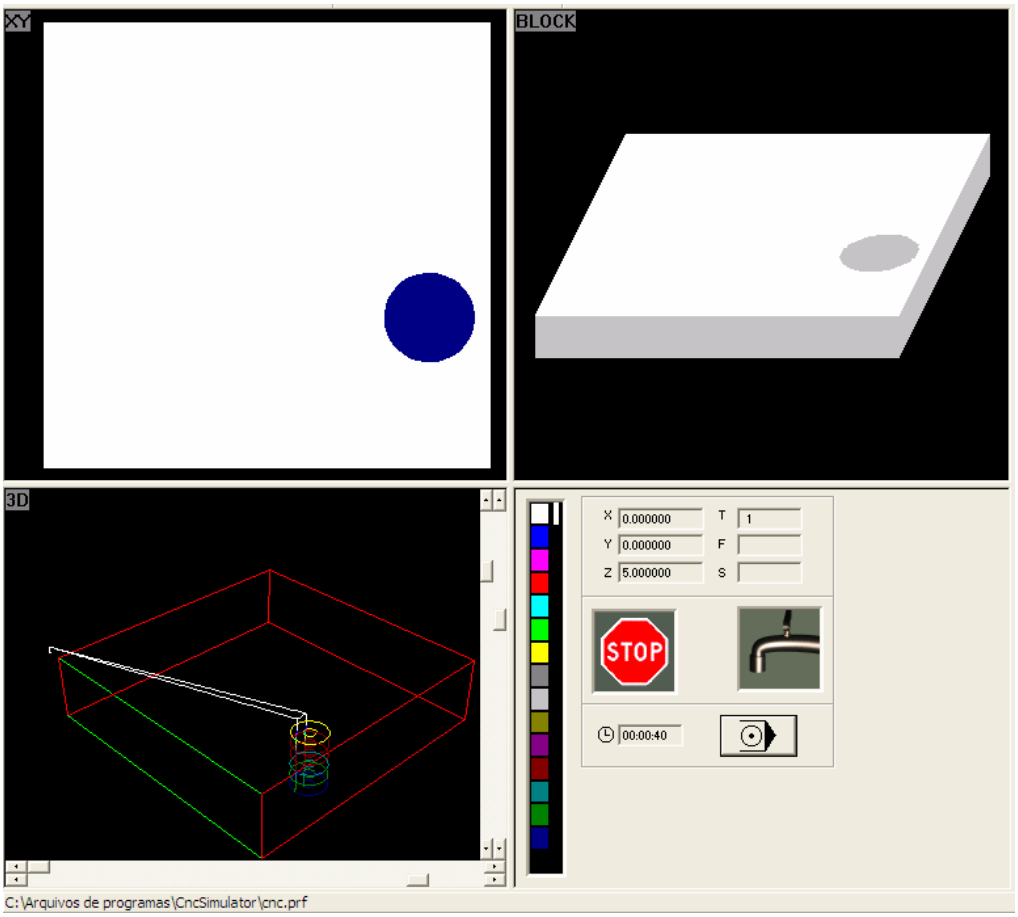
Visualização do código numérico agrupado dos modelos parciais 1 e 2.
Código numérico agrupado dos modelos parciais 3 e 4.

```
G01 X0 Y0
G01 Z 5
G01 X 25 Y 102.5
G01 Z-5
G02 X 25 Y 102.5 I 0 J 2.5
G01 Z 5
G01 X 25 Y 102.5
G01 Z-10
G02 X 25 Y 102.5 I 0 J 2.5
G01 Z 5
G01 X 25 Y 102.5
G01 Z-15
G02 X 25 Y 102.5 I 0 J 2.5
G01 Z 5
G01 X 25 Y 102.5
G01 Z-20
G02 X 25 Y 102.5 I 0 J 2.5
G01 Z 5
G01 X 25 Y 102.5
G01 Z-25
G02 X 25 Y 102.5 I 0 J 2.5
G01 Z 5
G01 X 25 Y 102.5
G01 Z-30
G02 X 25 Y 102.5 I 0 J 2.5
G01 Z 5
G01 X 105 Y 102.5
G01 Z-5
G02 X 105 Y 102.5 I 0 J 2.5
G01 Z 5
G01 X 105 Y 102.5
G01 Z-10
G02 X 105 Y 102.5 I 0 J 2.5
G01 Z 5
G01 X 105 Y 102.5
G01 Z-15
G02 X 105 Y 102.5 I 0 J 2.5
G01 Z 5
G01 X 105 Y 102.5
G01 Z-20
G02 X 105 Y 102.5 I 0 J 2.5
G01 Z 5
G01 X 105 Y 102.5
G01 Z-25
G02 X 105 Y 102.5 I 0 J 2.5
G01 Z 5
G01 X 105 Y 102.5
G01 Z-30
G02 X 105 Y 102.5 I 0 J 2.5
G01 Z 5
G01 X0 Y0 Z5
```



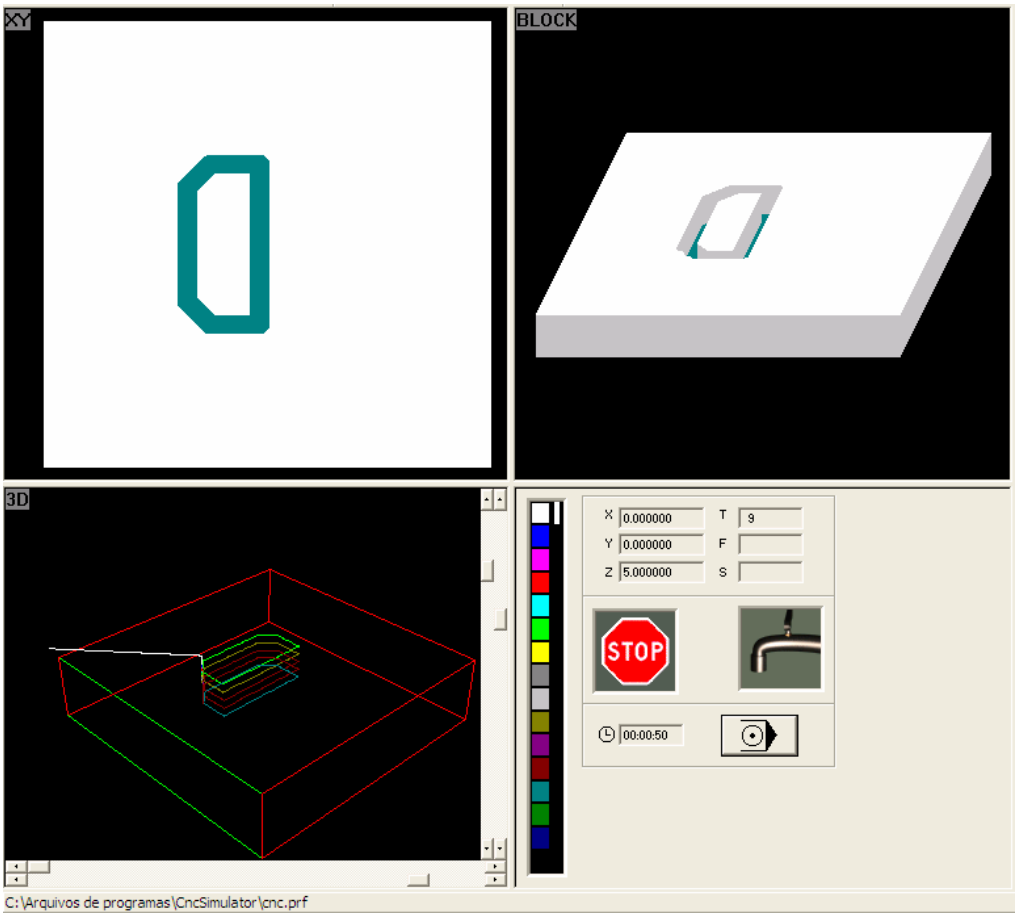
Visualização do código numérico agrupado dos modelos parciais 3 e 4.
Código numérico agrupado dos modelos parciais 5 e 6.

```
G01 X0 Y0
G01 Z 5
G01 X 110 Y 42.5
G01 Z-5
G02 X 110 Y 42.5 I 0 J 2.5
G01 Z 5
G01 X 110 Y 42.5
G01 Z-10
G02 X 110 Y 42.5 I 0 J 2.5
G01 Z 5
G01 X 110 Y 42.5
G01 Z-15
G02 X 110 Y 42.5 I 0 J 2.5
G01 Z 5
G01 X 110 Y 42.5
G01 Z-20
G02 X 110 Y 42.5 I 0 J 2.5
G01 Z 5
G01 X 110 Y 42.5
G01 Z-25
G02 X 110 Y 42.5 I 0 J 2.5
G01 Z 5
G01 X 110 Y 42.5
G01 Z-30
G02 X 110 Y 42.5 I 0 J 2.5
G01 Z 5
G01 X 110 Y 37.5
G01 Z-5
G02 X 110 Y 37.5 I 0 J 7.5
G01 Z 5
G01 X 110 Y 37.5
G01 Z-10
G02 X 110 Y 37.5 I 0 J 7.5
G01 Z 5
G01 X 110 Y 37.5
G01 Z-15
G02 X 110 Y 37.5 I 0 J 7.5
G01 Z 5
G01 X 110 Y 37.5
G01 Z-20
G02 X 110 Y 37.5 I 0 J 7.5
G01 Z 5
G01 X 110 Y 37.5
G01 Z-25
G02 X 110 Y 37.5 I 0 J 7.5
G01 Z 5
G01 X 110 Y 37.5
G01 Z-30
G02 X 110 Y 37.5 I 0 J 7.5
G01 Z 5
G01 X0 Y0 Z5
```



Visualização do código numérico agrupado dos modelos parciais 5 e 6.
Código numérico do modelo parcial 7

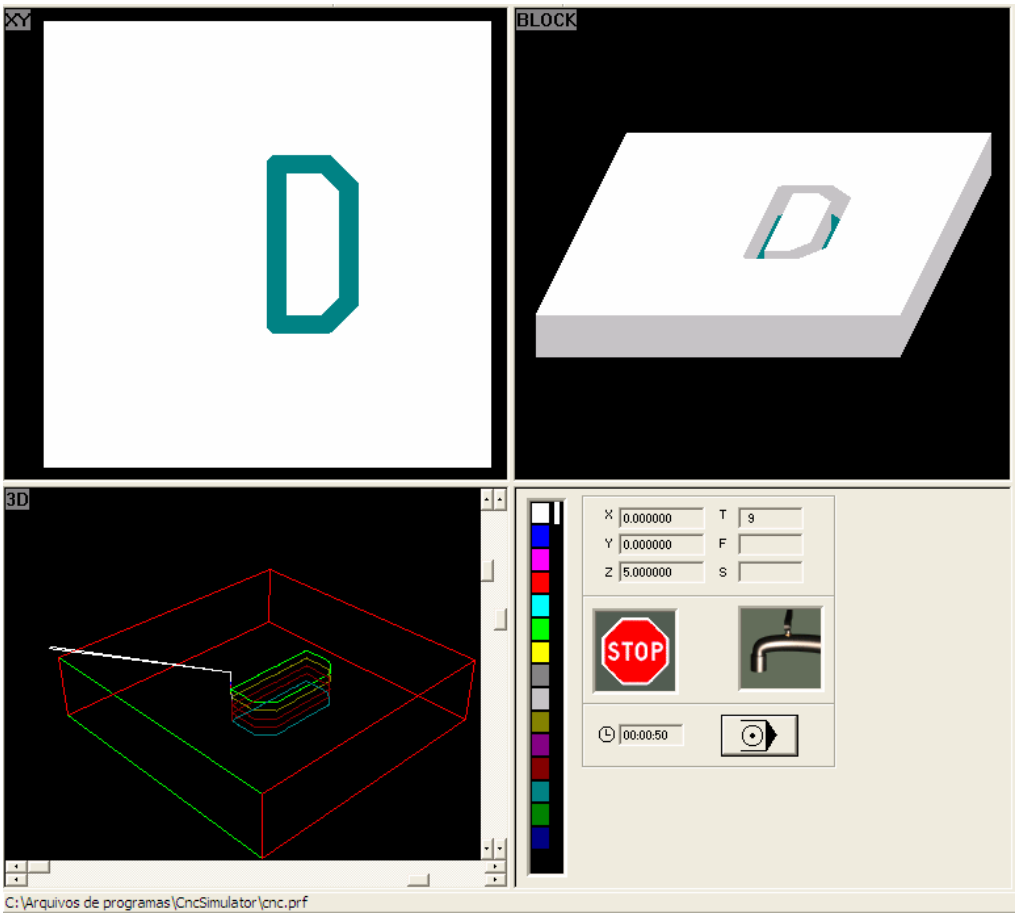
```
G01 X0 Y0
G01 Z 4
G01 X 49 Y 42.5
G01 Z-4
G01 X 62.5   Y 42.5
G01 X 62.5   Y 87.5
G01 X 49    Y 87.5
G01 X 42.5   Y 81
G01 X 42.5   Y 49
G01 X 49    Y 42.5
G01 Z 4
G01 X 49 Y 42.5
G01 Z-8
G01 X 62.5   Y 42.5
G01 X 62.5   Y 87.5
G01 X 49    Y 87.5
G01 X 42.5   Y 81
G01 X 42.5   Y 49
G01 X 49    Y 42.5
G01 Z 4
G01 X 49 Y 42.5
G01 Z-12
G01 X 62.5   Y 42.5
G01 X 62.5   Y 87.5
G01 X 49    Y 87.5
G01 X 42.5   Y 81
G01 X 42.5   Y 49
G01 X 49    Y 42.5
G01 Z 4
G01 X 49 Y 42.5
G01 Z-16
G01 X 62.5   Y 42.5
G01 X 62.5   Y 87.5
G01 X 49    Y 87.5
G01 X 42.5   Y 81
G01 X 42.5   Y 49
G01 X 49    Y 42.5
G01 Z 4
G01 X 49 Y 42.5
G01 Z-20
G01 X 62.5   Y 42.5
G01 X 62.5   Y 87.5
G01 X 49    Y 87.5
G01 X 42.5   Y 81
G01 X 42.5   Y 49
G01 X 49    Y 42.5
G01 Z 4
G01 X0 Y0
G01 Z5
```



Visualização do código numérico do modelo parcial 7

Código numérico do modelo parcial 8

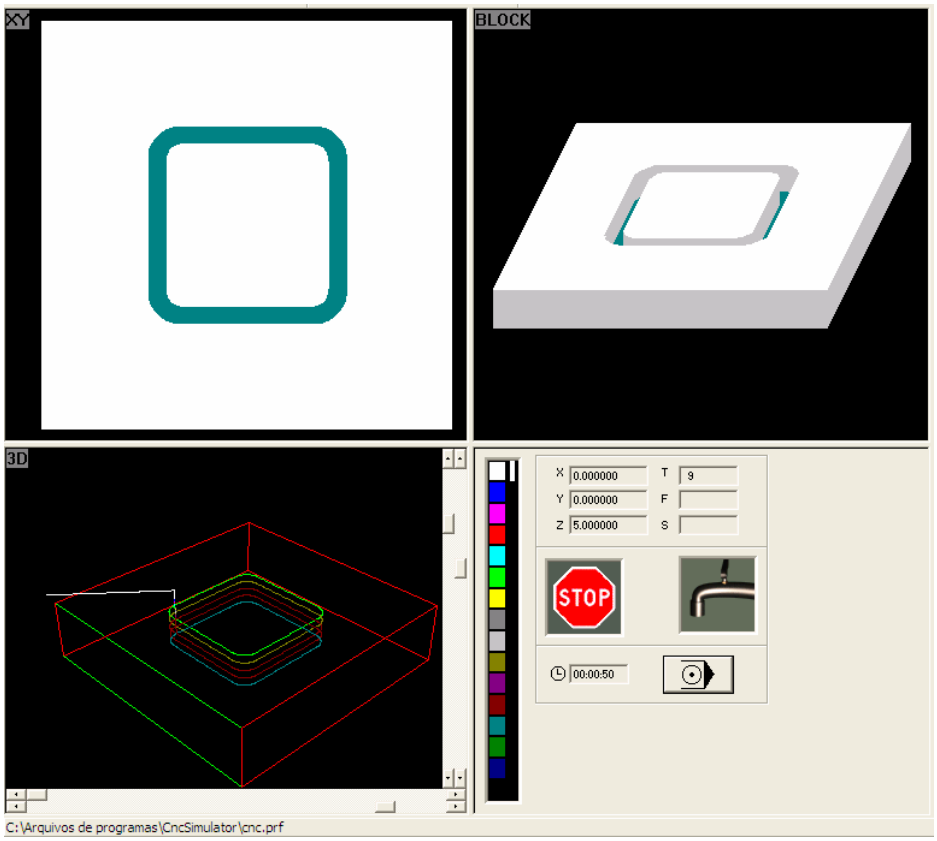
```
G01 X0 Y0
G01 Z 4
G01 X 67.5 Y 42.5
G01 Z-4
G01 X 81    Y 42.5
G01 X 87.5   Y 49
G01 X 87.5   Y 81
G01 X 81    Y 87.5
G01 X 67.5   Y 87.5
G01 X 67.5   Y 42.5
G01 Z 4
G01 X 67.5 Y 42.5
G01 Z-8
G01 X 81    Y 42.5
G01 X 87.5   Y 49
G01 X 87.5   Y 81
G01 X 81    Y 87.5
G01 X 67.5   Y 87.5
G01 X 67.5   Y 42.5
G01 Z 4
G01 X 67.5 Y 42.5
G01 Z-12
G01 X 81    Y 42.5
G01 X 87.5   Y 49
G01 X 87.5   Y 81
G01 X 81    Y 87.5
G01 X 67.5   Y 87.5
G01 X 67.5   Y 42.5
G01 Z 4
G01 X 67.5 Y 42.5
G01 Z-16
G01 X 81    Y 42.5
G01 X 87.5   Y 49
G01 X 87.5   Y 81
G01 X 81    Y 87.5
G01 X 67.5   Y 87.5
G01 X 67.5   Y 42.5
G01 Z 4
G01 X 67.5 Y 42.5
G01 Z-20
G01 X 81    Y 42.5
G01 X 87.5   Y 49
G01 X 87.5   Y 81
G01 X 81    Y 87.5
G01 X 67.5   Y 87.5
G01 X 67.5   Y 42.5
G01 Z 4
G01 X0 Y0 Z5
```



Visualização do código numérico do modelo parcial 8

Código numérico do modelo parcial 9

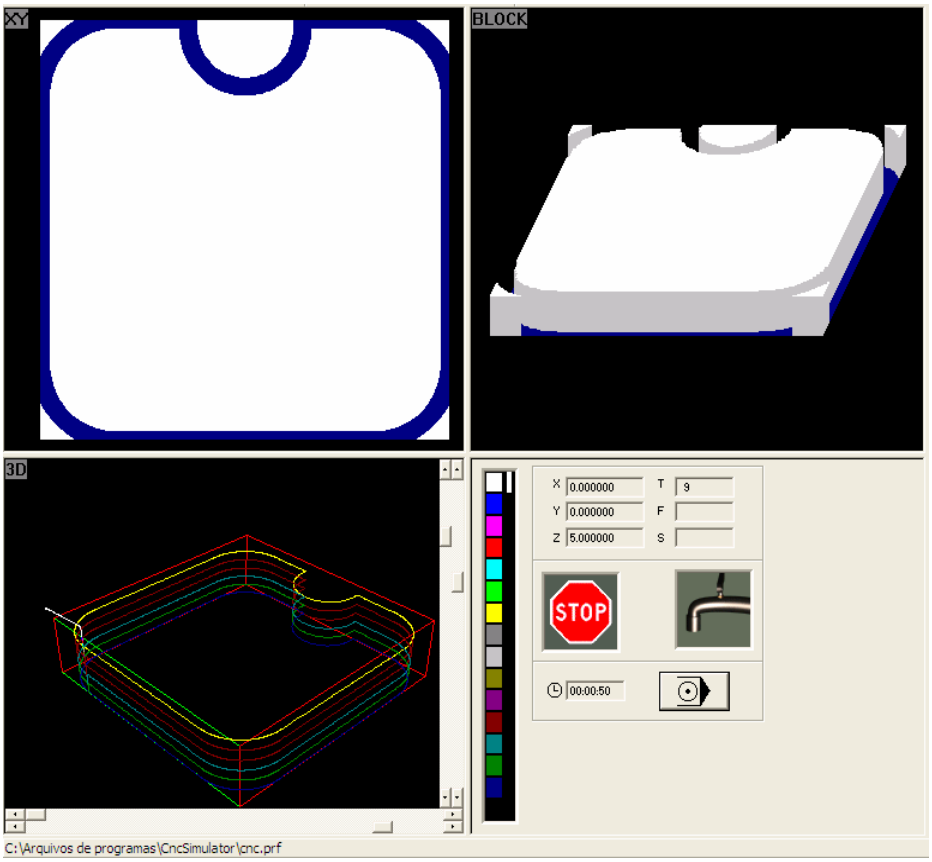
```
G01 X0 Y0
G01 Z 4
G01 X 37.5 Y 45
G01 Z-4
G03 X 45   Y 37.5 I 7.5 J 0
G01 X 85   Y 37.5
G03 X 92.5  Y 45 I 0 J 7.5
G01 X 92.5  Y 85
G03 X 85   Y 92.5 I -7.5 J 0
G01 X 45   Y 92.5
G03 X 37.5  Y 85 I 0 J -7.5
G01 X 37.5  Y 45
G01 Z 4
G01 X 37.5 Y 45
G01 Z-8
G03 X 45   Y 37.5 I 7.5 J 0
G01 X 85   Y 37.5
G03 X 92.5  Y 45 I 0 J 7.5
G01 X 92.5  Y 85
G03 X 85   Y 92.5 I -7.5 J 0
G01 X 45   Y 92.5
G03 X 37.5  Y 85 I 0 J -7.5
G01 X 37.5  Y 45
G01 Z 4
G01 X 37.5 Y 45
G01 Z-12
G03 X 45   Y 37.5 I 7.5 J 0
G01 X 85   Y 37.5
G03 X 92.5  Y 45 I 0 J 7.5
G01 X 92.5  Y 85
G03 X 85   Y 92.5 I -7.5 J 0
G01 X 45   Y 92.5
G03 X 37.5  Y 85 I 0 J -7.5
G01 X 37.5  Y 45
G01 Z 4
G01 Z-16
G03 X 45   Y 37.5 I 7.5 J 0
G01 X 85   Y 37.5
G03 X 92.5  Y 45 I 0 J 7.5
G01 X 92.5  Y 85
G03 X 85   Y 92.5 I -7.5 J 0
G01 X 45   Y 92.5
G03 X 37.5  Y 85 I 0 J -7.5
G01 X 37.5  Y 45
G01 Z 4
G01 Z-20
G03 X 45   Y 37.5 I 7.5 J 0
G01 X 85   Y 37.5
G03 X 92.5  Y 45 I 0 J 7.5
G01 X 92.5  Y 85
G03 X 85   Y 92.5 I -7.5 J 0
G01 X 45   Y 92.5
G03 X 37.5  Y 85 I 0 J -7.5
G01 X 37.5  Y 45
G01 Z 4
G01 X0 Y0
G01 Z5
```



Visualização do código numérico do modelo parcial 9

Código numérico do modelo parcial 10

```
G01 X 0 Y 0
G01 Z 5
G01 X 25 Y 2.5
G01 Z-5
G01 X 105   Y 2.5
G03 X 127.5  Y 25 I 0 J 22.5
G01 X 127.5  Y 105
G03 X 105   Y 127.5 I -22.5 J 0
G01 X 82.3   Y 127.5
G02 X 47.7  Y 127.5 I -17.3 J -2.5
G01 X 25    Y 127.5
G03 X 2.5   Y 105 I 0 J -22.5
G01 X 2.5   Y 25
G03 X 25    Y 2.5 I 22.5 J 0
G01 Z 5
G01 X 25 Y 2.5
G01 Z-10
G01 X 105   Y 2.5
G03 X 127.5  Y 25 I 0 J 22.5
G01 X 127.5  Y 105
G03 X 105 Y 127.5 I -22.5 J 0
G01 X 82.3   Y 127.5
G02 X 47.7 Y 127.5 I -17.3 J -2.5
G01 X 25    Y 127.5
G03 X 2.5   Y 105 I 0 J -22.5
G01 X 2.5   Y 25
G03 X 25    Y 2.5 I 22.5 J 0
G01 Z 5
G01 X 25 Y 2.5
G01 Z-15
G01 X 105   Y 2.5
G03 X 127.5  Y 25 I 0 J 22.5
G01 X 127.5  Y 105
G03 X 105   Y 127.5 I -22.5 J 0
G01 X 82.3   Y 127.5
G02 X 47.7   Y 127.5 I -17.3 J -2.5
G01 X 25    Y 127.5
G03 X 2.5   Y 105 I 0 J -22.5
G01 X 2.5   Y 25
G03 X 25    Y 2.5 I 22.5 J 0
G01 Z 5
G01 X 25 Y 2.5
G01 Z-20
G01 X 105   Y 2.5
G03 X 127.5  Y 25 I 0 J 22.5
G01 X 127.5  Y 105
G03 X 105   Y 127.5 I -22.5 J 0
G01 X 82.3   Y 127.5
G02 X 47.7   Y 127.5 I -17.3 J -2.5
G01 X 25    Y 127.5
G03 X 2.5   Y 105 I 0 J -22.5
G01 X 2.5   Y 25
G03 X 25    Y 2.5 I 22.5 J 0
G01 Z 5
G01 X 25 Y 2.5
G01 Z-25
G01 X 105   Y 2.5
G03 X 127.5  Y 25 I 0 J 22.5
G01 X 127.5  Y 105
G03 X 105   Y 127.5 I -22.5 J 0
G01 X 82.3   Y 127.5
G02 X 47.7   Y 127.5 I -17.3 J -2.5
G01 X 25    Y 127.5
G03 X 2.5   Y 105 I 0 J -22.5
G01 X 2.5   Y 25
G03 X 25    Y 2.5 I 22.5 J 0
G01 Z 5
G01 X 25 Y 2.5
G01 Z-30
G01 X 105   Y 2.5
G03 X 127.5  Y 25 I 0 J 22.5
G01 X 127.5  Y 105
G03 X 105 Y 127.5 I -22.5 J 0
G01 X 82.3   Y 127.5
G02 X 47.7   Y 127.5 I -17.3 J -2.5
G01 X 25    Y 127.5
G03 X 2.5   Y 105 I 0 J -22.5
G01 X 2.5   Y 25
G03 X 25    Y 2.5 I 22.5 J 0
G01 Z 5
G01 X 25 Y 2.5
G01 Z 5
G01 X 0 Y 0 Z 5
```

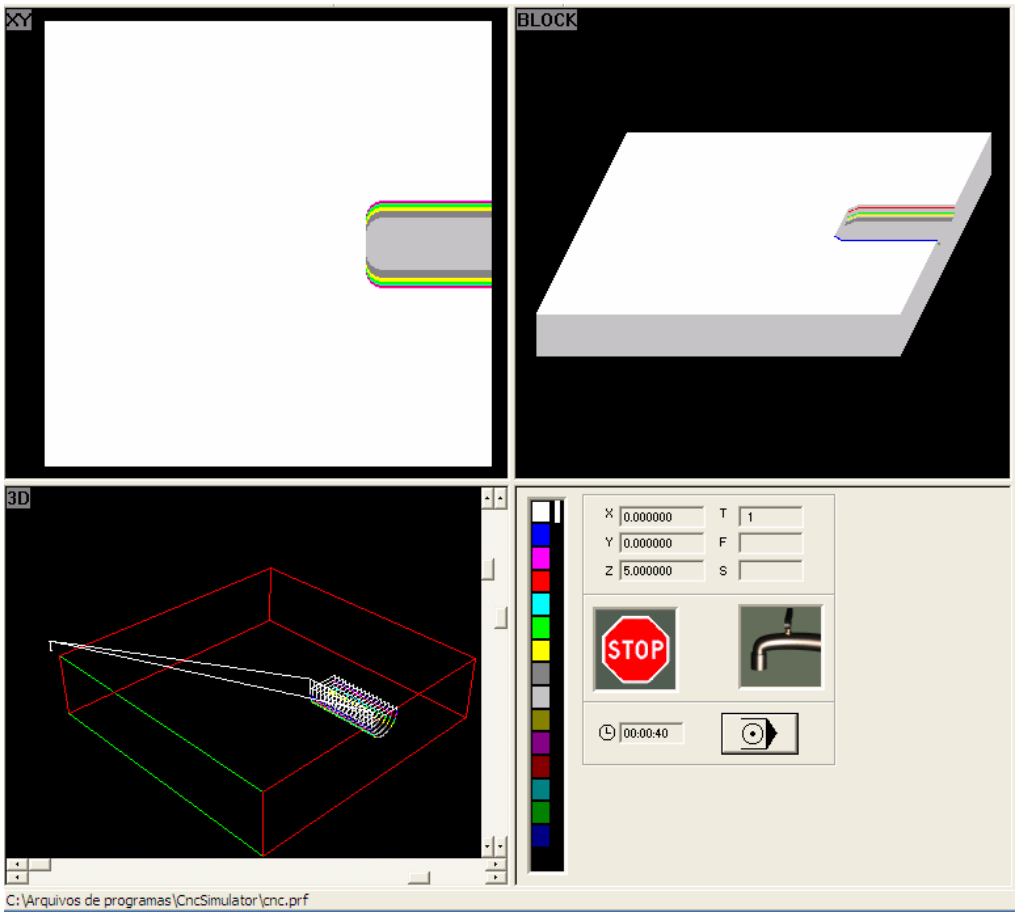


Visualização do código numérico do modelo parcial 10

Código numérico do modelo parcial 11

```
G01 X0 Y0
G01 Z5
G01 Z 7
G01 X 97 Y 57.5
G01 Z 2
G01 Y 57.5 Z 2
G01 Y 72.5  Z 2
G01 Y 72.5  Z 0
G01 Y 72.1  Z-2.3
G01 Y 71.1  Z-4.4
G01 Y 69.4  Z-6.1
G01 Y 67.3  Z-7.1
G01 Y 65   Z-7.5
G01 Y 62.7  Z-7.1
G01 Y 60.6  Z-6.1
G01 Y 58.9  Z-4.4
G01 Y 57.9  Z-2.3
G01 Y 57.5  Z 0
G01 Y 57.5  Z 2
G01 Z 7
G01 X 99 Y 57.5
G01 Z 2
G01 Z 2
G01 Y 57.5 Z 2
G01 Y 72.5  Z 2
G01 Y 72.5  Z 0
G01 Y 72.1  Z-2.3
G01 Y 71.1  Z-4.4
G01 Y 69.4  Z-6.1
G01 Y 67.3  Z-7.1
G01 Y 65   Z-7.5
G01 Y 62.7  Z-7.1
G01 Y 60.6  Z-6.1
G01 Y 58.9  Z-4.4
G01 Y 57.9  Z-2.3
G01 Y 57.5  Z 0
G01 Y 57.5  Z 2
G01 Z 7
G01 X 101 Y 57.5
G01 Z 2
G01 Z 2
G01 Y 57.5 Z 2
G01 Y 72.5  Z 2
G01 Y 72.5  Z 0
G01 Y 72.1  Z-2.3
G01 Y 71.1  Z-4.4
G01 Y 69.4  Z-6.1
G01 Y 67.3  Z-7.1
G01 Y 65   Z-7.5
G01 Y 62.7  Z-7.1
G01 Y 60.6  Z-6.1
G01 Y 58.9  Z-4.4
G01 Y 57.9  Z-2.3
G01 Y 57.5  Z 0
G01 Y 57.5  Z 2
G01 Z 7
G01 X 103 Y 57.5
G01 Z 2
G01 Z 2
G01 Y 57.5 Z 2
G01 Y 72.5  Z 2
G01 Y 72.5  Z 0
G01 Y 72.1  Z-2.3
G01 Y 71.1  Z-4.4
G01 Y 69.4  Z-6.1
G01 Y 67.3  Z-7.1
G01 Y 65   Z-7.5
G01 Y 62.7  Z-7.1
G01 Y 60.6  Z-6.1
G01 Y 58.9  Z-4.4
G01 Y 57.9  Z-2.3
G01 Y 57.5  Z 0
G01 Y 57.5  Z 2
G01 Z 7
G01 X 105 Y 57.5
G01 Z 2
G01 Z 2
G01 Y 57.5 Z 2
G01 Y 72.5  Z 2
G01 Y 72.5  Z 0
G01 Y 72.1  Z-2.3
G01 Y 71.1  Z-4.4
G01 Y 69.4  Z-6.1
G01 Y 67.3  Z-7.1
G01 Y 65   Z-7.5
G01 Y 62.7  Z-7.1
G01 Y 60.6  Z-6.1
G01 Y 58.9  Z-4.4
G01 Y 57.9  Z-2.3
G01 Y 57.5  Z 0
G01 Y 57.5  Z 2
G01 Z 7
G01 X 107 Y 57.5
G01 Z 2
G01 Z 2
G01 Y 57.5 Z 2
G01 Y 72.5  Z 2
G01 Y 72.5  Z 0
G01 Y 72.1  Z-2.3
G01 Y 71.1  Z-4.4
G01 Y 69.4  Z-6.1
G01 Y 67.3  Z-7.1
G01 Y 65   Z-7.5
G01 Y 62.7  Z-7.1
G01 Y 60.6  Z-6.1
G01 Y 58.9  Z-4.4
G01 Y 57.9  Z-2.3
G01 Y 57.5  Z 0
G01 Y 57.5  Z 2
G01 Z 7
G01 X 109 Y 57.5
G01 Z 2
G01 Z 2
G01 Y 57.5 Z 2
G01 Y 72.5  Z 2
G01 Y 72.5  Z 0
G01 Y 72.1  Z-2.3
G01 Y 71.1  Z-4.4
G01 Y 69.4  Z-6.1
G01 Y 67.3  Z-7.1
G01 Y 65   Z-7.5
G01 Y 62.7  Z-7.1
G01 Y 60.6  Z-6.1
G01 Y 58.9  Z-4.4
G01 Y 57.9  Z-2.3
G01 Y 57.5  Z 0
G01 Y 57.5  Z 2
G01 Z 7
G01 X 111 Y 57.5
G01 Z 2
G01 Z 2
G01 Y 57.5 Z 2
G01 Y 72.5  Z 2
G01 Y 72.5  Z 0
G01 Y 72.1  Z-2.3
G01 Y 71.1  Z-4.4
G01 Y 69.4  Z-6.1
G01 Y 67.3  Z-7.1
G01 Y 65   Z-7.5
G01 Y 62.7  Z-7.1
G01 Y 60.6  Z-6.1
G01 Y 58.9  Z-4.4
G01 Y 57.9  Z-2.3
G01 Y 57.5  Z 0
G01 Y 57.5  Z 2
G01 Z 7
G01 X 113 Y 57.5
G01 Z 2
G01 Z 2
G01 Y 57.5 Z 2
G01 Y 72.5  Z 2
G01 Y 72.5  Z 0
G01 Y 72.1  Z-2.3
G01 Y 71.1  Z-4.4
G01 Y 69.4  Z-6.1
G01 Y 67.3  Z-7.1
G01 Y 65   Z-7.5
G01 Y 62.7  Z-7.1
G01 Y 60.6  Z-6.1
G01 Y 58.9  Z-4.4
G01 Y 57.9  Z-2.3
G01 Y 57.5  Z 0
G01 Y 57.5  Z 2
G01 Z 7
G01 X 115 Y 57.5
G01 Z 2
G01 Z 2
G01 Y 57.5 Z 2
G01 Y 72.5  Z 2
G01 Y 72.5  Z 0
G01 Y 72.1  Z-2.3
G01 Y 71.1  Z-4.4
G01 Y 69.4  Z-6.1
G01 Y 67.3  Z-7.1
G01 Y 65   Z-7.5
G01 Y 62.7  Z-7.1
G01 Y 60.6  Z-6.1
```

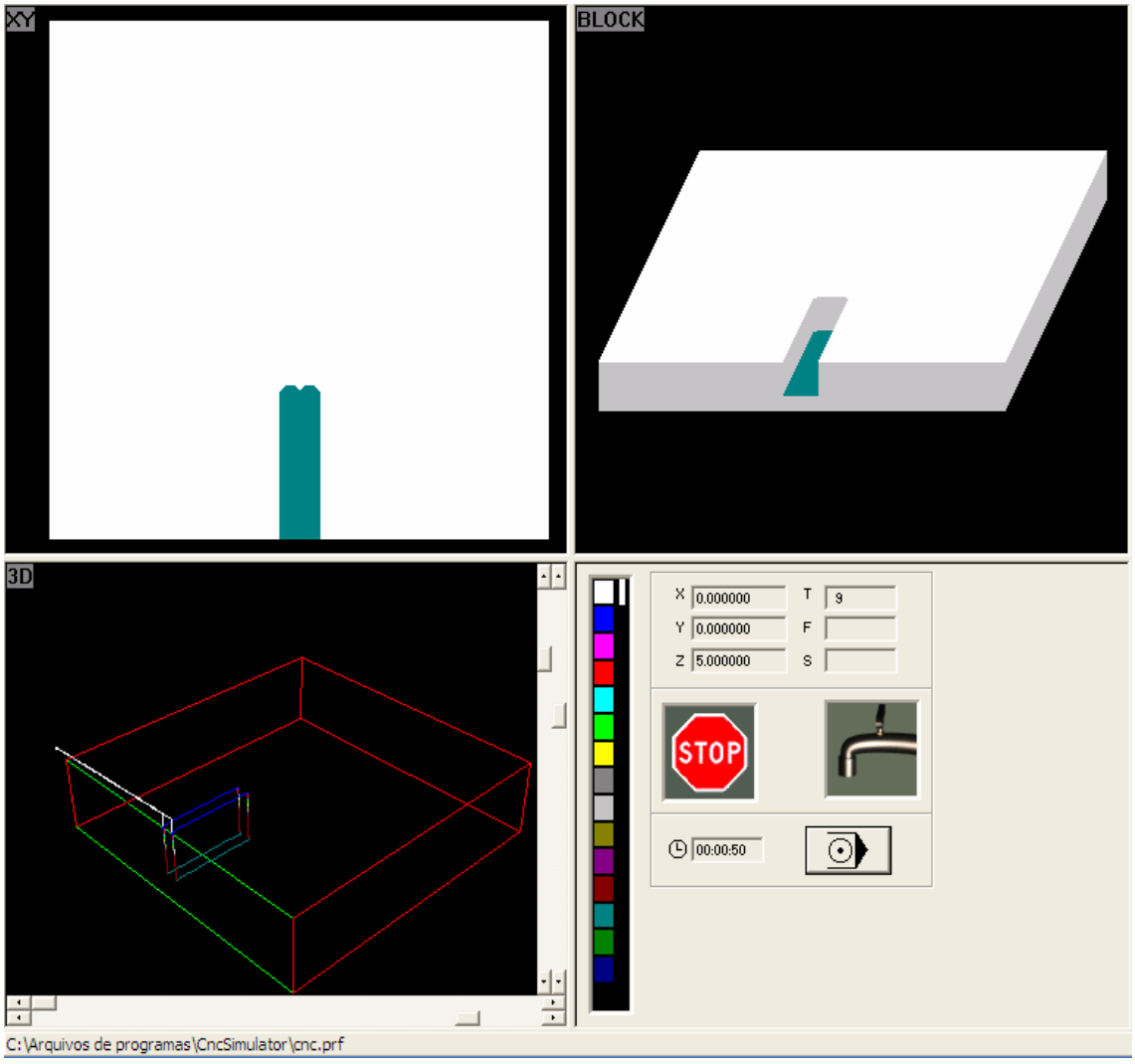
```
G01 Y 58.9   Z-4.4
G01 Y 57.9   Z-2.3
G01 Y 57.5   Z 0
G01 Y 57.5   Z 2
G01 Z 7
G01 X 117 Y 57.5
G01 Z 2
G01 Z 2
G01 Y 57.5 Z 2
G01 Y 72.5   Z 2
G01 Y 72.5   Z 0
G01 Y 72.1   Z-2.3
G01 Y 71.1   Z-4.4
G01 Y 69.4   Z-6.1
G01 Y 67.3   Z-7.1
G01 Y 65    Z-7.5
G01 Y 62.7   Z-7.1
G01 Y 60.6   Z-6.1
G01 Y 58.9   Z-4.4
G01 Y 57.9   Z-2.3
G01 Y 57.5   Z 0
G01 Y 57.5   Z 2
G01 Z 7
G01 X 119 Y 57.5
G01 Z 2
G01 Z 2
G01 Y 57.5 Z 2
G01 Y 72.5   Z 2
G01 Y 72.5   Z 0
G01 Y 72.1   Z-2.3
G01 Y 71.1   Z-4.4
G01 Y 69.4   Z-6.1
G01 Y 67.3   Z-7.1
G01 Y 65    Z-7.5
G01 Y 62.7   Z-7.1
G01 Y 60.6   Z-6.1
G01 Y 58.9   Z-4.4
G01 Y 57.9   Z-2.3
G01 Y 57.5   Z 0
G01 Y 57.5   Z 2
G01 Z 7
G01 X 121 Y 57.5
G01 Z 2
G01 Z 2
G01 Y 57.5 Z 2
G01 Y 72.5   Z 2
G01 Y 72.5   Z 0
G01 Y 72.1   Z-2.3
G01 Y 71.1   Z-4.4
G01 Y 69.4   Z-6.1
G01 Y 67.3   Z-7.1
G01 Y 65    Z-7.5
G01 Y 62.7   Z-7.1
G01 Y 60.6   Z-6.1
G01 Y 58.9   Z-4.4
G01 Y 57.9   Z-2.3
G01 Y 57.5   Z 0
G01 Y 57.5   Z 2
G01 Z 7
G01 X 123 Y 57.5
G01 Z 2
G01 Z 2
G01 Y 57.5 Z 2
G01 Y 72.5   Z 2
G01 Y 72.5   Z 0
G01 Y 72.1   Z-2.3
G01 Y 71.1   Z-4.4
G01 Y 69.4   Z-6.1
G01 Y 67.3   Z-7.1
G01 Y 65    Z-7.5
G01 Y 62.7   Z-7.1
G01 Y 60.6   Z-6.1
G01 Y 58.9   Z-4.4
G01 Y 57.9   Z-2.3
G01 Y 57.5   Z 0
G01 Y 57.5   Z 2
G01 Z 7
G01 X 125 Y 57.5
G01 Z 2
G01 Z 2
G01 Y 57.5 Z 2
G01 Y 72.5   Z 2
G01 Y 72.5   Z 0
G01 Y 72.1   Z-2.3
G01 Y 71.1   Z-4.4
G01 Y 69.4   Z-6.1
G01 Y 67.3   Z-7.1
G01 Y 65    Z-7.5
G01 Y 62.7   Z-7.1
G01 Y 60.6   Z-6.1
G01 Y 58.9   Z-4.4
G01 Y 57.9   Z-2.3
G01 Y 57.5   Z 0
G01 Y 57.5   Z 2
G01 Z 7
G01 X 127 Y 57.5
G01 Z 2
G01 Z 2
G01 Y 57.5 Z 2
G01 Y 72.5   Z 2
G01 Y 72.5   Z 0
G01 Y 72.1   Z-2.3
G01 Y 71.1   Z-4.4
G01 Y 69.4   Z-6.1
G01 Y 67.3   Z-7.1
G01 Y 65    Z-7.5
G01 Y 62.7   Z-7.1
G01 Y 60.6   Z-6.1
G01 Y 58.9   Z-4.4
G01 Y 57.9   Z-2.3
G01 Y 57.5   Z 0
G01 Y 57.5   Z 2
G01 Z 7
G01 X 129 Y 57.5
G01 Z 2
G01 Z 2
G01 Y 57.5 Z 2
G01 Y 72.5   Z 2
G01 Y 72.5   Z 0
G01 Y 72.1   Z-2.3
G01 Y 71.1   Z-4.4
G01 Y 69.4   Z-6.1
G01 Y 67.3   Z-7.1
G01 Y 65    Z-7.5
G01 Y 62.7   Z-7.1
G01 Y 60.6   Z-6.1
G01 Y 58.9   Z-4.4
G01 Y 57.9   Z-2.3
G01 Y 57.5   Z 0
G01 Y 57.5   Z 2
G01 Z 7
G01 X 0 Y 0
G01 Z 7
G01 Z5
```



Visualização do código numérico do modelo parcial 11

Código numérico do modelo parcial 12

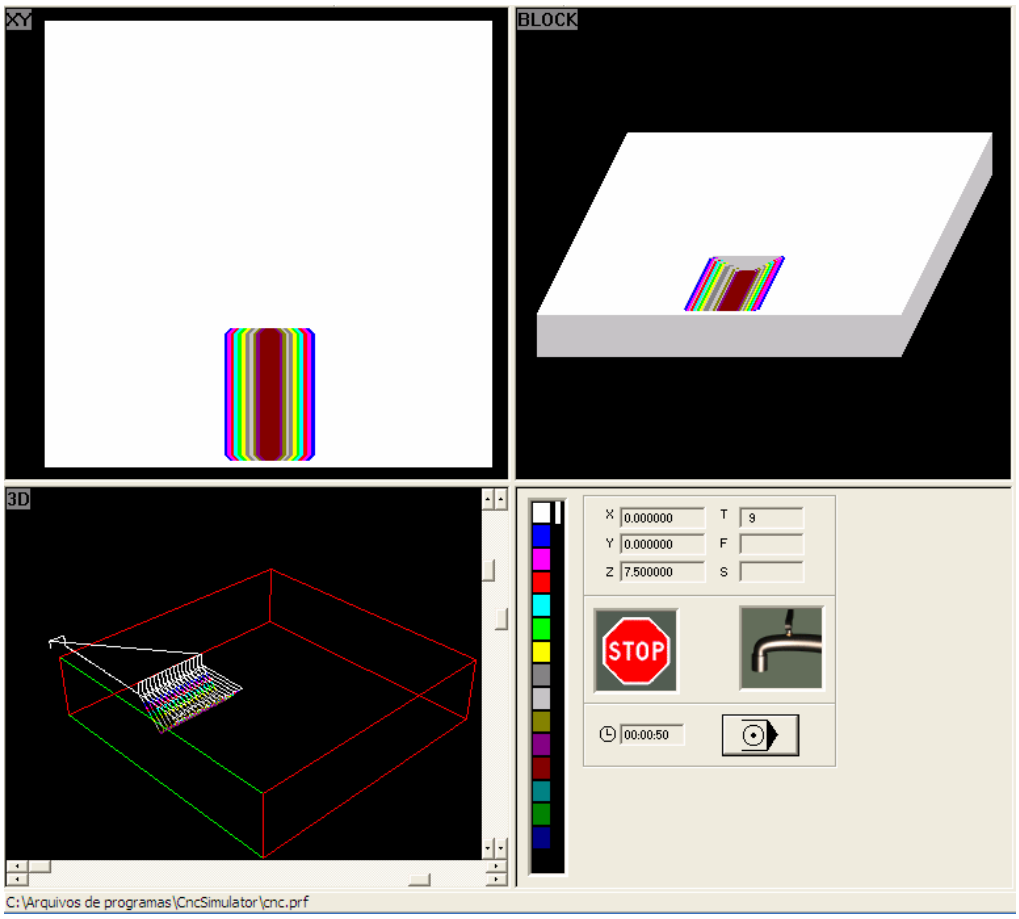
G01 Z 5
G01 X 62.5 Y 3
G01 Z 0
G01 Y 3 Z 0
G01 Y 37 Z 0
G01 Y 37 Z-20
G01 Y 3 Z-20
G01 Y 3 Z 0
G01 Z 5
G01 X 67.5 Y 3
G01 Z 0
G01 Y 3 Z 0
G01 Y 37 Z 0
G01 Y 37 Z-20
G01 Y 3 Z-20
G01 Y 3 Z 0
G01 Z 5
G01 X 0 Y 0
G01 Z 5



Visualização do código numérico do modelo parcial 12

Código numérico do modelo parcial 13

```
G01 Z 7.5
G01 Y 7
G01 Z 2.5
G01 Z 2.5
G01 X 52.5 Z 2.5
G01 X 77.5 Z 2.5
G01 X 65 Z-10
G01 X 52.5 Z 2.5
G01 Z 7.5
G01 Y 7
G01 X 52.5 Y 9
G01 Z 2.5
G01 Z 2.5
G01 Z 2.5
G01 X 52.5 Z 2.5
G01 X 77.5 Z 2.5
G01 X 65 Z-10
G01 X 52.5 Z 2.5
G01 Z 7.5
G01 Y 9
G01 X 52.5 Y 11
G01 Z 2.5
G01 Z 2.5
G01 Z 2.5
G01 X 52.5 Z 2.5
G01 X 77.5 Z 2.5
G01 X 65 Z-10
G01 X 52.5 Z 2.5
G01 Z 7.5
G01 Y 11
G01 X 52.5 Y 13
G01 Z 2.5
G01 Z 2.5
G01 Z 2.5
G01 X 52.5 Z 2.5
G01 X 77.5 Z 2.5
G01 X 65 Z-10
G01 X 52.5 Z 2.5
G01 Z 7.5
G01 Y 13
G01 X 52.5 Y 15
G01 Z 2.5
G01 Z 2.5
G01 Z 2.5
G01 X 52.5 Z 2.5
G01 X 77.5 Z 2.5
G01 X 65 Z-10
G01 X 52.5 Z 2.5
G01 Z 7.5
G01 Y 15
G01 X 52.5 Y 17
G01 Z 2.5
G01 Z 2.5
G01 Z 2.5
G01 X 52.5 Z 2.5
G01 X 77.5 Z 2.5
G01 X 65 Z-10
G01 X 52.5 Z 2.5
G01 Z 7.5
G01 Y 17
G01 X 52.5 Y 19
G01 Z 2.5
G01 Z 2.5
G01 Z 2.5
G01 X 52.5 Z 2.5
G01 X 77.5 Z 2.5
G01 X 65 Z-10
G01 X 52.5 Z 2.5
G01 Z 7.5
G01 Y 19
G01 X 52.5 Y 21
G01 Z 2.5
G01 Z 2.5
G01 Z 2.5
G01 X 52.5 Z 2.5
G01 X 77.5 Z 2.5
G01 X 65 Z-10
G01 X 52.5 Z 2.5
G01 Z 7.5
G01 Y 21
G01 X 52.5 Y 23
G01 Z 2.5
G01 Z 2.5
G01 Z 2.5
G01 X 52.5 Z 2.5
G01 X 77.5 Z 2.5
G01 X 65 Z-10
G01 X 52.5 Z 2.5
G01 Z 7.5
G01 Y 23
G01 X 52.5 Y 25
G01 Z 2.5
G01 Z 2.5
G01 Z 2.5
G01 X 52.5 Z 2.5
G01 X 77.5 Z 2.5
G01 X 65 Z-10
G01 X 52.5 Z 2.5
G01 Z 7.5
G01 Y 25
G01 X 52.5 Y 27
G01 Z 2.5
G01 Z 2.5
G01 Z 2.5
G01 X 52.5 Z 2.5
G01 X 77.5 Z 2.5
G01 X 65 Z-10
G01 X 52.5 Z 2.5
G01 Z 7.5
G01 Y 27
G01 X 52.5 Y 29
G01 Z 2.5
G01 Z 2.5
G01 Z 2.5
G01 X 52.5 Z 2.5
G01 X 77.5 Z 2.5
G01 X 65 Z-10
G01 X 52.5 Z 2.5
G01 Z 7.5
G01 Y 29
G01 X 52.5 Y 31
G01 Z 2.5
G01 Z 2.5
G01 Z 2.5
G01 X 52.5 Z 2.5
G01 X 77.5 Z 2.5
G01 X 65 Z-10
G01 X 52.5 Z 2.5
G01 Z 7.5
G01 Y 31
G01 X 52.5 Y 33
G01 Z 2.5
G01 Z 2.5
G01 Z 2.5
G01 X 52.5 Z 2.5
G01 X 77.5 Z 2.5
G01 X 65 Z-10
G01 X 52.5 Z 2.5
G01 Z 7.5
G01 Y 33
G01 X 52.5 Y 35
G01 Z 2.5
G01 Z 2.5
G01 Z 2.5
G01 X 52.5 Z 2.5
G01 X 77.5 Z 2.5
G01 X 65 Z-10
G01 X 52.5 Z 2.5
G01 Z 7.5
G01 Y 35
G01 X 52.5 Y 37
G01 Z 2.5
G01 Z 2.5
G01 Z 2.5
G01 X 52.5 Z 2.5
G01 X 77.5 Z 2.5
G01 X 65 Z-10
G01 X 52.5 Z 2.5
G01 Z 7.5
G01 Y 37
G01 X 52.5 Y 39
G01 Z 2.5
G01 Z 2.5
G01 Z 2.5
G01 X 52.5 Z 2.5
G01 X 77.5 Z 2.5
G01 X 65 Z-10
G01 X 52.5 Z 2.5
G01 Z 7.5
G01 X 0 Y 0
G01 Z 7.5
M30
```



Visualização do código numérico do modelo parcial 13