**Universidade Estadual de Maringá**
**Centro de Tecnologia**
**Departamento de Engenharia de Produção**

# A Contribuição de Software de Modelagem Tridimensional no Auxílio e Desenvolvimento de *Layout* Industrial

*Bruno Edson dos Santos*

**TCC-EP-12-2010**

**Maringá - Paraná**
**Brasil**

Universidade Estadual de Maringá
Centro de Tecnologia
Departamento de Engenharia de Produção

# A Contribuição de Software de Modelagem Tridimensional no Auxílio e Desenvolvimento de *Layout* Industrial

*Bruno Edson dos Santos*

**TCC-EP-12-2010**

Trabalho de graduação apresentado como requisito de avaliação no curso de graduação em Engenharia de Produção na Universidade Estadual de Maringá – UEM.
Orientador: Prof. Dr. Dante Alves Medeiros Filho
Co-orientadora: Prof.(ª): Carla Fernanda Marek Gasparini

**Maringá - Paraná**
**2010**

RESUMO

É crescente o uso de ferramentas de computação gráfica na concepção, projeto e análise nas áreas da pesquisa e além. Assim, considera-se que haja demanda da engenharia de produção por esses tipos de ferramentas com a intenção de melhorar a experiência de desenvolvimento de *layout* industrial. Por se tratar de uma pesquisa de natureza aplicada com caráter experimental, foram pesquisados alguns algoritmos e técnicas de construção de *layout* para serem inseridos em um software de modelagem e animação voltado inicialmente a produções artísticas. Para auxiliar na implementação dos algoritmos, foi revisado alguns conceitos de engenharia de *software*, dentre eles o reuso de software, além da necessária consulta à documentação do *software*. No presente trabalho foi demonstrado o procedimento para incorporar os algoritmos no software, assim como, realizado uma série de considerações sobre o *software*, os algoritmos e o fluxo de trabalho.

**Palavras-chave:** *Layout*, projeto de arranjo físico, computação gráfica, modelagem tridimensional.

## ABSTRACT

*An increasing use of computer graphics tools in the concept, analysis and design in the areas of research and beyond. Thus, it is considered that there is a demand of industrial engineering for these types of tools with the intention of improving the development experience of the industrial layout. Because it is an applied research with an experimental nature, have been investigated some algorithms and techniques for building layout to be inserted into a modeling and animation software focused initially on artistic productions. To assist in the implementation of algorithms, has been revisited some concepts of software engineering, among them, besides the need to reuse refers to the software documentation. It is shown how to proceed to incorporate the software algorithms. Finally it was done a number of considerations about the software, algorithms and on the workflow.*

Keyword: layout, layout design, computer graphics, three-dimensional modeling.

SUMÁRIO

# LISTA DE ILUSTRAÇÕES

# LISTA DE TABELAS

# LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| AGA | Algoritmos Genéticos de Agrupamento |
| AG | Algoritmos Genéticos |
| API | *Aplication Programming Interface* |
| CRAFT | *Computerized Relative Allocation of Facilities Technique* |
| DCA | *Direct Clustering Algorithm* (Algoritmo de Agrupamento Direto) |
| HGA | *Hybrid Genetic Algorithm* (Algoritmo Genético Híbrido) |
| LOGIC | *Layout Optimization with Guillotine Induced Cuts* |
| MEL | *Maya Embedded Language* |
| MST | *Maximum Spanning Tree* |
| NaN | Not a Number (Subsidiária da Neo-Geo) |
| OO | Orientação a Objetos |
| OOP | *Object-Oriented Programming* (Programação Orientada a Objetos) |
| PFA | *Production Flow Analisys* (Análise de Fluxo de Produção) |
| QAP | *Quadratic Assignment Problem* (Problema da Atribuição Quadrática) |
| ROC | *Rank Order Clustering* (Agrupamento por ordem de ranking) |

# 1 INTRODUÇÃO

Os novos paradigmas de sistemas de manufatura exigem instalações industriais flexíveis e com capacidade de lidar com uma razoável quantidade de produtos ainda que dentro de uma mesma linha. O aumento da complexidade do projeto de produto e da programação da produção levou a diferentes soluções de *layout*s de produção industrial. Quatro tipos são extensamente abordados na literatura: *layout* por processo, por produto, de manufatura celular e por posição fixa. Gaither & Frezier (2002) e Martins & Laugeni (2005) anotam ainda a possibilidade de adoção de *layout*s híbridos ou combinados.

As características dos métodos de avaliação e desenvolvimento de *layout*s industriais podem subsidiar a construção de algoritmos computacionais, e com isso, surge oportunidade de serem criadas soluções de software para a análise e auxílio no desenvolvimento de arranjos físicos. Junto com a solução em termos de algoritmos, a moderna computação está começando a disponibilizar soluções em modelos tridimensionais. A importância da modelagem 3D está na interface mais amigável, na facilidade de manipular os objetos, que inseridos em software de modelagem 3D são representações quase fiéis das estações de trabalho reais.

Cada vez mais tecnologias são aplicadas a sistemas produtivos almejando dotar a manufatura de melhor capacidade de absorver as decorrentes demandas de mercado. Essas tecnologias visam em geral à integração de toda a empresa ou parte da mesma. Na área de projeto da instalação industrial as questões estratégicas de mercado e questões de planejamento tático devem e consideradas. O que foi exposto até o momento é para denotar a dificuldade de se chegar ao arranjo físico ideal para a área disponível que muitas vezes a primeira vista aparenta ser insuficiente para a estratégia ou tática de produção da empresa.

## 1.1 Justificativa

Nesse contexto, é inegável que no projeto de instalações industriais questões não numéricas e restrições de custos de projeto ou até restrições de políticas existam e são impactantes. O auxílio da computação e até da computação gráfica se dá na redução de custos de projeto, na facilidade de visualização das opções de arranjos físico, na agilidade proporcionada pelo

conjunto hardware e software nos cálculos e procedimentos repetitivos e na liberação de tempo para o engenheiro ou projetista pensar e ponderar as questões estratégicas, políticas, táticas.

A modelagem para análise de um arranjo físico em software gráfico tridimensional apresenta um custo de recursos e tempo grande. O engenheiro ou projetista que se dispõe ao trabalho irá se deparar com a necessidade de construir no software as máquinas, equipamentos e estações de trabalho manualmente, um trabalho que é de tão longa duração quanto a necessidade de tornar os modelos tridimensionais próximos a uma máquina ou estação de trabalho real. Ainda na questão do nível de realismo pretendido pelo projetista, este realismo irá necessitar de incremento na capacidade do hardware sob o qual esteja sendo realizado o projeto. Deverá fazer deslocamentos desses modelos tridimensionais através do cenário virtual, muitas vezes uma estação de cada vez, até encontrar o posicionamento que considera bom, porém sem as ferramentas para medições e automatização, porque estes não fazem parte do escopo de muitos aplicativos de modelagem e animação voltados à produção artística. E na questão de ajustes e deslocamentos a necessidade de capacidade de hardware será proporcional ao tamanho do arranjo físico e do detalhe dos modelos tridimensionais que estão contidos nesse arranjo.

Por outro lado, tem-se variados aplicativos voltados para desenho técnico, porém boa parte desses aplicativos são para desenhos em duas dimensões e apenas alguns poucos contém ferramentas de análise de *layout* implementadas e prontas para uso.

Atualmente existem várias técnicas e ferramentas computacionais e algoritmos otimizantes e heurísticos. Segundo Torres (2001) há três tipos de ferramentas muito utilizadas ainda no desenvolvimento de projetos de plantas industriais:

a) **Ferramentas CAD** – *(Computer Aided Design)* – usadas para representações de plantas dos projetos e representações tridimensionais para análise arquitetônica. Dada a sua difusão, hoje há disposição vários software que adicionam funções para análise e design de arranjos físicos;
b) **Ferramentas de Simulação de Sistemas de Eventos Discretos** – softwares que vem evoluindo, e são aplicados para análise de processos produtivos e programação da

produção. Com esses sistemas especialistas, podem ser avaliados os efeitos da programação da produção sobre o *layout* e vice-versa;

c) **Ferramentas de Simulação Humana** – softwares idealizados para analisar o elemento humano em seu posto de trabalho, visando tornar os objetos e procedimentos mais adaptados ao homem, reduzindo os problemas de interação. A análise dos postos de trabalho e integração entre centros de produção são indispensáveis para o desenvolvimento do *layout*.

O problema a ser solucionado é como fazer as atuais ferramentas de modelagem tridimensional propor automaticamente arranjos de máquinas e pessoas na fábrica e concomitantemente posicionar a proposta no projeto arquitetônico da própria fábrica, assim como efetuar o árduo trabalho de posicionamento geral, deixando para o engenheiro ou projetista a tarefa de efetuar ajustes finos na proposta do *layout* industrial, produzindo assim ganho de esforço financeiro, humano e ganho de tempo.

Nesse projeto, está fora do escopo, a simulação de sistemas de eventos discretos e a simulação humana.

## 1.2 Objetivo

O objetivo e também a motivação do presente trabalho é investigar e avaliar a possibilidade de tornar os aplicativos de modelagem tridimensional voltado para produções artísticas em aplicativos voltados as necessidades de engenheiros de produção ou projetistas de instalações industriais. Isso através do uso da engenharia de software e programação na construção de componentes e ferramentas.

O motivo de se voltar atenção para aplicações 3D é a facilidade intuitiva que esses aplicativos proporcionam na manipulação e visualização do problema do *layout* industrial.

# 2 REVISÃO DA LITERATURA

Neste capítulo será abordado toda a literatura referente a tipos de arranjos físicos básicos encontrados na indústria, alguns dos vários algoritmos de ordenação dos arranjos físicos assim como os *software* de modelagem tridimensional.

## 2.1 Arranjo Físico da Fábrica

Os projetos de arranjos físico das fábricas são processos complexos visto como decisão estratégica, ou seja, com foco no longo prazo e geralmente designados a engenheiros, também chamados de projetistas de arranjos físico, cujo trabalho é definir um novo *layout* ou efetuar melhorias no mesmo. Um bom projeto de arranjo físico pode garantir um sistema de manufatura mais enxuto e eficiente, importantes fatores para garantir um diferencial competitivo à empresa frente ao mercado (ARGOUD, 2007).

Por essa e outras razões, nos últimos 50 anos, vem sendo discutidos e desenvolvidos diversos tipos de arranjo físico adequado a um ou outro processo produtivo, métodos e algoritmos para auxiliar na análise e formação desses arranjos e modelos e processos para auxiliar o projetista em seu trabalho.

Nesse capítulo será revista a literatura sobre os tipos básicos de arranjos físicos bem como os processos, modelos e algoritmos utilizados e suas dificuldades e os softwares de modelagem tridimensional.

## 2.2 Tipos Tradicionais de Arranjo Físico

Na literatura diversos autores apresentam conceitos de diversos tipos de arranjos físicos, estes com suas respectivas vantagens e desvantagens assim como forma de aplicação. A grande maioria é derivada de arranjos físicos básicos ou até uma combinação destes (MARTINS E LAUGENI, 2005; DAVIS E AQUILANO, 2001; ARGOUD, 2007, GAITHER E FRAZIER, 2002). São quatro os arranjos físicos básicos:

a) Arranjo físico posicional;
b) Arranjo físico por processo ou funcional;
c) Arranjo físico por produto;
d) Arranjo físico celular.

Os quatro tipos são apresentados nessa seção.

### 2.2.1 Arranjo Físico Posicional

O produto permanece fixo, enquanto os recursos, mão de obra, máquinas e ferramentas vão até ele para se realizar o processamento. É muito flexível porque pode incorporar mudanças de projeto e entre projetos. A demanda é baixa em termos de volume e o produto é difícil de ser movimentado. Este é o caso da construção de navios, plataformas, edifícios, aviões de e equipamentos de grande porte. A Figura 1 apresenta o diagrama de um arranjo posicional (ARGOUD, 2007).

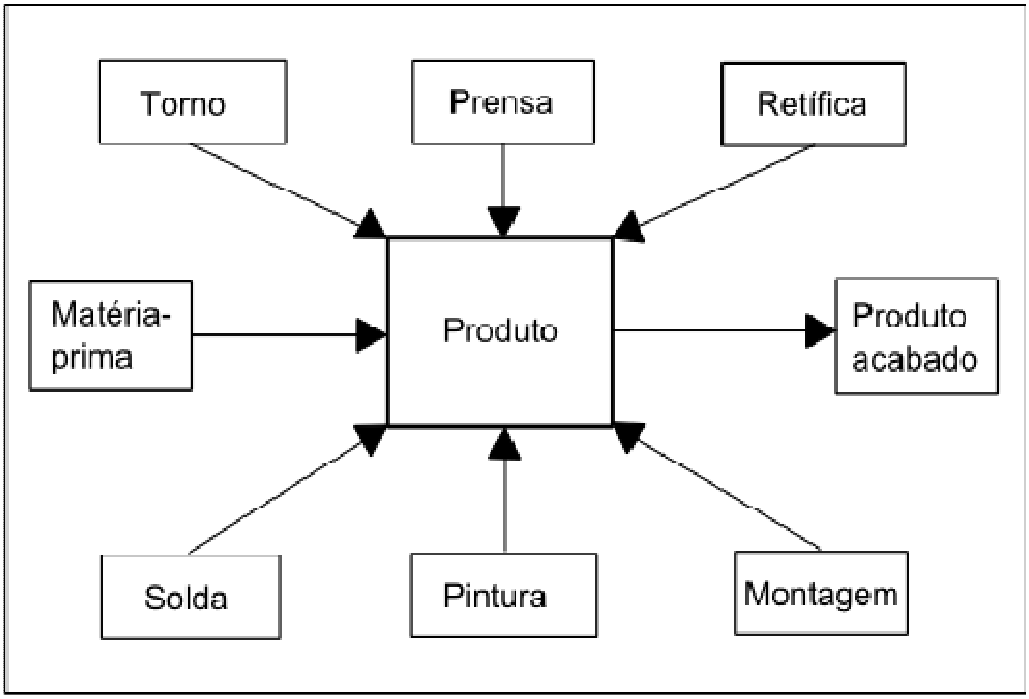


**Figura 1: Arranjo físico posicional.**
**Fonte: Argoud (2007).**

Na literatura referenciada, apenas Slack et al (2009) apresenta um procedimento para a formação desse tipo de arranjo físico, que é dividido em seis passos, e busca definir a forma de instalação e como os possíveis locais onde os recursos serão centralizados, fazendo uma análise baseada em um conjunto de critérios de avaliação para determinar a localização de cada um desses locais em torno do produto.

### 2.2.2 Arranjo Físico por Produto

As máquinas são arranjadas de acordo com uma seqüência de operações. Conforme apresentado na Figura 2. O manuseio de materiais e o controle do processo ficam mais facilitados. Porém é o que requer equipamentos específicos de alto rendimento e alterações no produto implicam em mudança na disposição das máquinas e isso acaba por torná-lo de baixa flexibilidade. A maioria das tarefas é simples e repetitiva e não necessita de mão de obra altamente qualificada. A movimentação de materiais é reduzida, os *leadtimes* são curtos e o inventário é baixo. Os equipamentos precisam ser altamente confiáveis pois uma parada em qualquer um deles pode parar toda a linha. A linha de montagem é o exemplo mais típico de tipo de arranjo físico (ARGOUD, 2007).

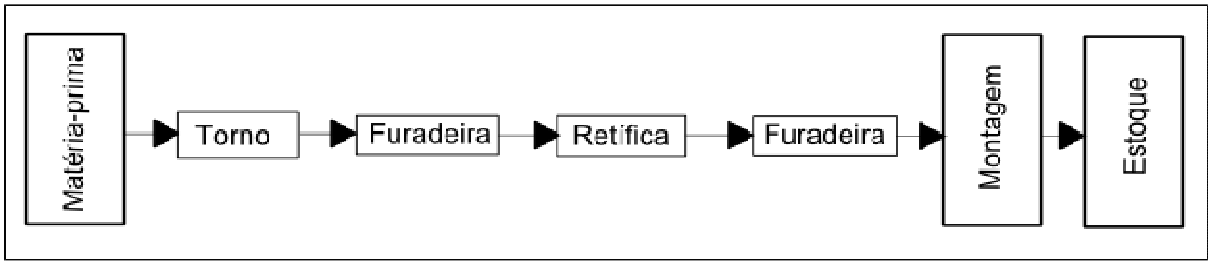


**Figura 1: Arranjo físico por produto.**
**Fonte: Argoud (2007).**

Esse tipo de arranjo físico é encontrado em fábricas de produtos de larga escala como sapatos, eletrodomésticos, bebidas, etc. Slack et al (2009) cita outros exemplos como:

a) Montagem de automóveis: uma seqüência de processos é utilizada para cada modelo;
b) Programas de vacinação em massa: todos os convocados requerem uma mesma seqüência de atividades, como identificação, vacinação e aconselhamento;

c) Restaurante *self-service*: os clientes passam por diversas seções, tais como frios, saladas, prato principal, sobremesa geralmente dispostos em linhas.

A maior preocupação nesse tipo de arranjo físico está relacionado ao balanço de linha, cujo objetivo é a otimização da seqüência de operações de modo a reduzir o número de estações de trabalho e ociosidade nas tarefas. Na literatura pode-se encontrar vários métodos e algoritmos para isso. Ghosh e Gagnon (*apud* TIBERTI, 2003, p12) fizeram uma revisão sobre o balanceamento de linha e discutiram vários métodos baseados em algoritmos como o CONSOAL, CLAB, MALB, NULISP e MUST.

### 2.2.3 Arranjo físico por processo ou funcional

O arranjo físico por processo, também chamado de funcional se caracteriza pelo agrupamento, num mesmo espaço chamado de departamento ou seção, de máquinas e equipamentos que desempenham funções semelhantes. Desse modo o fluxo de materiais ocorrerá então de uma seção para a outra para que as operações necessárias sejam realizadas (ARGOUD, 2007).

Como produtos diferentes possuem diferentes necessidades, acabam por demandar diferentes roteiros e rotas entre esses departamentos, criando padrões de fluxo bastante complexos. É um tipo de arranjo físico que primeiro surgiu, junto com a revolução industrial, e é a forma mais antiga de organização de recursos numa produção. O arranjo funcional está representador na Figura 3.

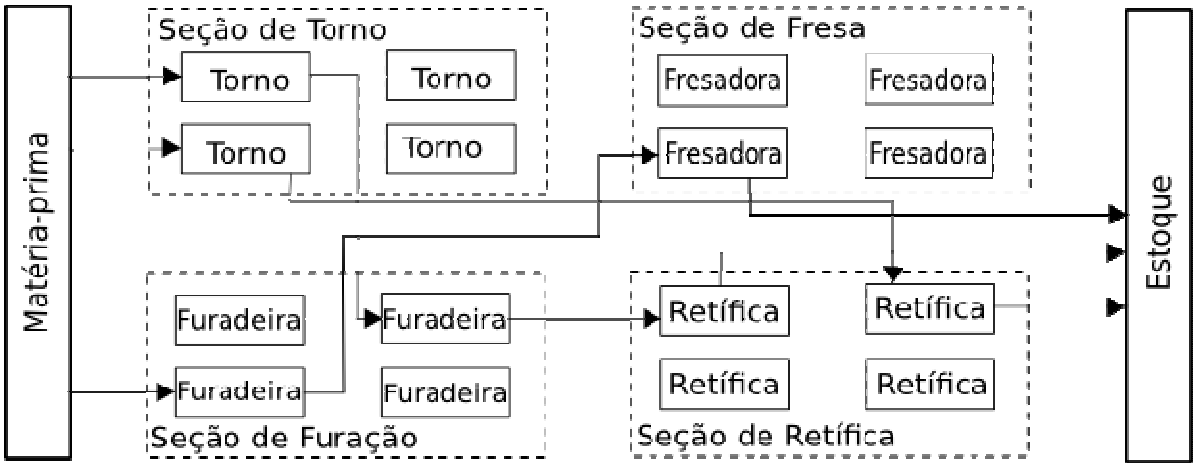


**Figura 2: Arranjo físico por processo.**

**Fonte: Tiberti (2003).**

Esse tipo de arranjo físico pode e é encontrado no chão de fábrica sob a forma de seções, como seção de pintura, torno, montagem, soldagem e etc. Também na parte administrativa das indústrias, como departamentos de compras, vendas, atendimento a cliente, suporte técnico e etc.

Os métodos e algoritmos empregados no projeto desse tipo de arranjo físico são conhecidos sob dois tipos, de construção ou de melhoria de *layout* de blocos ou de planta.

a) *Métodos e algoritmos de construção* usam apenas os dados para a formação de arranjos físicos para definir novos *layouts* de bloco, ignorando a existência de qualquer arranjo pré-existente;
b) Métodos e algoritmos de melhoria toma o *layout* de blocos existente para a otimização desse *layout*.

Por ser o mais antigo, há muitos algoritmos computacionais para a sua construção e melhoria proposto por vários autores. Os mais antigos e conhecidos são: *Computerized Relative Allocation of Facilities Technique* (CRAFT), BLOCPLAN, *Layout Optimization with Guillotine Induced Cuts* (LOGIC) TOMPKINS et al (*apud* TIBERTI, 2003, p12) e *Maximum Spanning Tree* (MST) BAASE (*apud* TIBERTI, 2003, p12). Esses algoritmos em geral objetivam minimizar o fluxo ou então maximizar as adjacência entre os diversos departamentos. Em trabalhos mais recentes, novas tecnologias vêm sendo usadas para o desenvolvimento desse tipo de *layout*, como algoritmos genético híbrido (HGA) proposto por Lee e Lee (*apud* TIBERTI, 2003, p12) para tentar resolver o problema de arranjo físico de departamentos com áreas diferentes (TIBERTI, 2003).

### 2.2.4 Arranjo físico Celular

Neste tipo de arranjo físico, as máquinas são arranjadas em grupos de tipos diversos, destinadas a atender não mais um determinado produto, mas de uma família de peças. Segundo Lorini (1993) é o tipo de *layout* com maior tendência de adoção na atualidade pela engenharia industrial, que busca, no suprimento dos processos necessários, a produção de determinadas peças, uma especialização por componentes quando se determina o arranjo. Caracteriza-se então, pela diversificação dos produtos, uma família de peças com diferentes tamanhos de lotes.

Com relação a outros tipos de arranjos físicos, diminuem consideravelmente os tempos improdutivos, como os de movimentação e manuseio de peças, além dos *setups* de máquinas. A vantagem dos arranjos em linha e funcional é que estão todas as máquinas estão próximas uma das outras, na seqüência mais adequada ao processo, ao mesmo tempo em que se tem a flexibilidade de execução, não mais para um só produto, mas uma família deles. Na Figura 4 é ilustrado o modelo de arranjo físico celular.

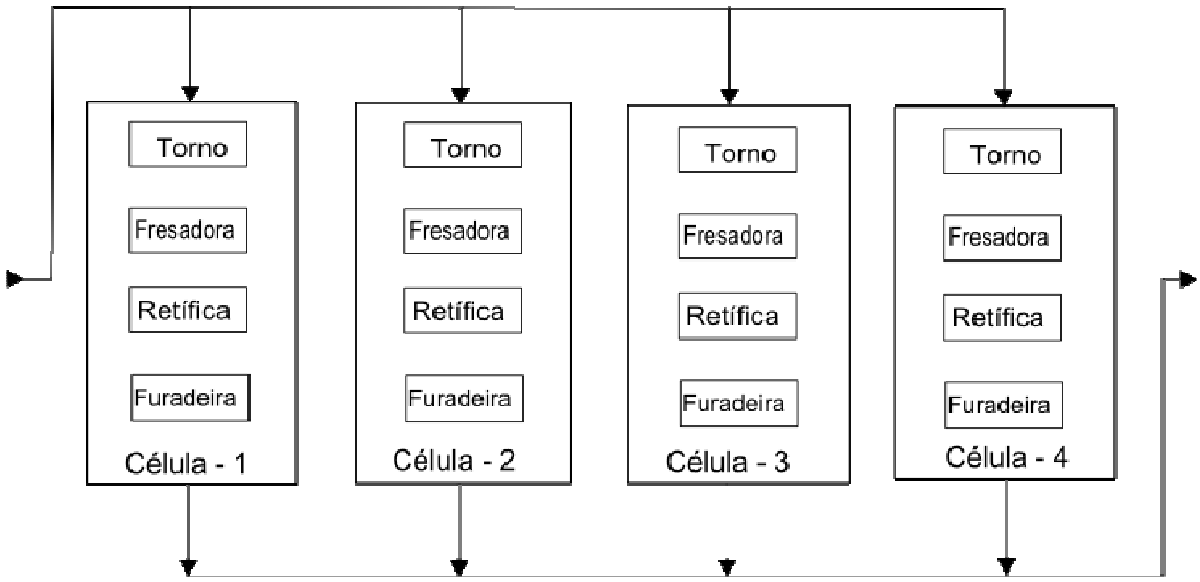


**Figura 3: Arranjo físico celular. Adaptado de Tiberti (2003).**

Vale a pena salientar a importância da diferença com o arranjo físico por processo. Isso porque no arranjo celular existe um alto fluxo intradepartamental e um baixo fluxo interdepartamental. No geral, grande parte do fluxo de materiais de uma peça se dava dentro da célula, mas algumas peças podem necessitar de operações em outras células. Ainda sim, como aponta Gonçalves Filho (*apud* TIBERTI, 2003, p14), o padrão do fluxo se aproxima do padrão de fluxo de arranjo físico por produto.

Devido a larga utilização desse tipo de arranjo, existe uma extensa literatura sobre os métodos e algoritmos em seus projetos. A codificação e classificação, análise de fluxo de produção (PFA – *Production Flow Analisys*), técnicas de agrupamento, algoritmos heurísticos e modelos matemáticos. Muitos algoritmos foram criados utilizando variadas técnicas como agrupamentos baseados em lógica matemática e heurística, como o DCA (*Direct Clustering Algorithm*), o ROC (*Rank Order Clustering*), coeficiente de similaridade (KUSIAK e CHO, 1992) e K-Means (IRANI et al, 2000), até algoritmos genéticos ADEL e EL-BAZ (2004), o algoritmos genético de agrupamento AGA (ARGOUD, 2007). As técnicas e algoritmos desse tipo de arranjo físico serão melhores tratadas.

## 2.3 Algoritmos de ordenação de arranjos físicos

Nesta seção serão elencados e detalhados alguns algoritmos de ordenação principalmente para arranjo físico do tipo celular. O caso do arranjo por produto, geralmente é um arranjo em linha ou de modo mais elaborado, arranjo em U é razoavelmente simples a modelagem. A grande questão nesse tipo de arranjo é o balanceamento de linha. Em arranjo físico de processo, a abordagem será dada sobre o usuário projetista do *layout*, ou seja, buscar-se-á, dispor ferramentas para projetar o *layout* de um modo mais manual, ou seja, inicialmente sem auxílio de algoritmos para automatização da tarefa.

### 2.3.1 ROC (*Rank Order Clustering*)

Esse algoritmo foi bem abordado por Lorini (1993) porém foi desenvolvido e apresentado por KING (*apud* LORINI, 1993) e posteriormente aperfeiçoado por King e Narkornchai como a versão ROC2. Na concepção original, o método estabelece uma matriz binária, onde “1” representa a necessidade de uma peça na máquina e “0”, o caso oposto. Um valor decimal

correspondente a cada palavra binária é atribuído para linhas e colunas. Com estes valores, o algoritmo ROC rearranja as linhas e colunas da matriz de modo iterativo, até que as linhas e colunas fiquem dispostas em ordem decrescente.

Para melhor ilustração do método, Lorini (1993) exemplifica o método conforme é ilustrado na tabela 1. Primeiramente estabelece-se o respectivo valor de cada linha e sua posição correspondente a uma ordem de grandeza decrescente dos valores.

$$(1 * 2^3) + (0 * 2^2) + (1 * 2^1) + (0 * 2^0) = 10 \quad \text{(equação 1)}$$

**Tabela 1: Matriz máquina - peça**

| **Peso** | **2^5** | **2^4** | **2^3** | **2^2** | **2^1** | **2^0** | | |
|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | **Valor** | **Ordem** |
| **1** | | | 1 | | 1 | | 10 | 5 |
| **2** | | 1 | 1 | | | | 24 | 4 |
| **3** | 1 | | | 1 | | | 36 | 2 |
| **4** | | 1 | 1 | | 1 | | 26 | 3 |
| **5** | 1 | | | 1 | | 1 | 37 | 1 |

**Fonte: Lorini (1993)**

Alterando-se a posição das linhas, de modo que os valores fiquem posicionados em ordem decrescente, passe a calcular o valor para as colunas.

**Tabela 2: Linhas ordenadas decrescente**

| | **1** | **2** | **3** | **4** | **5** | **6** | |
|---|---|---|---|---|---|---|---|
| **5** | 1 | | | 1 | | | |
| **3** | 1 | | | 1 | | 1 | |
| **4** | | 1 | 1 | | 1 | | |
| **2** | | 1 | 1 | | | | |
| **1** | | | 1 | | 1 | | |
| | 24 | 6 | 7 | 24 | 5 | 16 | **Valor** |
| | 1 | 5 | 4 | 2 | 6 | 3 | **Ordem** |

**Fonte: Lorini (1993)**

Alterando-se a posição das colunas para ordem decrescente, e repetindo-se o processo, se necessário, até que todas as linhas e colunas estejam ordenadas, resulta em uma matriz classificada.

**Tabela 2: Matriz ordenada**

| | **1** | **4** | **6** | **3** | **2** | **5** | **Valor** | **Ordem** |
|---|---|---|---|---|---|---|---|---|
| **5** | 1 | 1 | 1 | | | | 56 | 1 |
| **3** | 1 | 1 | 1 | | | | 48 | 2 |
| **4** | | | | 1 | 1 | 1 | 7 | 3 |
| **2** | | | | 1 | 1 | | 6 | 4 |
| **1** | | | | 1 | | 1 | 5 | 5 |
| **Valor** | 24 | 24 | 16 | 7 | 6 | 5 | | |
| **Ordem** | 1 | 2 | 3 | 4 | 5 | 6 | | |

**Fonte: Lorini (1993)**

Obtém-se assim, por meio de um ordenamento decrescente dos valores decimais calculados, tanto para linhas e colunas, o agrupamento das peças em famílias:

- família 1 (peças 1 -4 -6) para a célula 1 (máquinas 5 -3)
- família 2 (peças 3 -2 -5) para a célula 2 (máquinas 1 -2 -4)

### 2.3.2 Otimização com Busca Tabu

Proposto por Martins et al (2003), a busca Tabu age como um algoritmo de melhoria, portanto, necessita de uma solução inicial. O desafio enfrentado nesse algoritmo é o problema de atribuição quadrática (QAP – *Quadratic Assignment Problem*). O QAP pode ser interpretado da seguinte forma: suponha que m departamentos precisam ser alocados em n localizações, de modo que, a cada localização j, seja atribuído um único departamento i. Para representar a interação entre os departamentos, um peso positivo é associado a cada par de departamentos. Portanto, o problema consiste em atribuir um departamento a uma localização diferente dos demais de modo a minimizar a soma dos pesos vezes as distâncias entre eles.

Nessa abordagem foram consideradas várias restrições como:

a) *Quadro externo (dimensões da fábrica)*: o espaço destinado a conter os departamentos é retangular. E todos os departamentos devem estar contidos nesse espaço;
b) *Área*: os departamentos apresentam exigências individuais de área;
c) *Formatos*: os departamentos apresentam formatos retangulares ou quadrados;
d) *Orientação*: livre (vertical ou horizontal);
e) *Razão de aspecto*: a razão de aspecto ai de um departamento i é dada pelo comprimento C do departamento i pela sua largura L. Departamentos com razão de aspecto variável permitem melhor flexibilidade retangular;

f) *Sobreposição*: em ambientes de piso único os departamentos não podem se sobrepor;

g) *Fixos e móvel*: departamentos "fixos" a priori possuem localização definida. Os móveis podem ocupar qualquer local ainda não utilizado;

h) *Percentual de redução*: um local disponível à atribuição de um departamento qualquer não poderá sofrer variação de área além do limite;

i) *Áreas "mortas"*: é comum nos ambientes industriais haver áreas impedidas de serem utilizadas (pilares, escadas, elevadores, etc);

j) Adjacência entre departamentos: em muitos casos práticos os departamentos precisam ficar adjacentes, com no mínimo dois pontos de fronteira em comum. Em outros casos, devem ficar afastados;

k) Adjacência entre o departamento e a fronteira do ambiente industrial: em alguns casos os departamentos podem ficar junto à fronteira do ambiente ou não;

l) Distância: duas formas de medir a distância: retangular e euclidiana. Se atribuirmos um departamento i na localização $Pi = (xi, yi)$e o departamento j na localização$Pj = (xj, yj)$, então podemos definir a distância entre os departamentos i e j como por.

$$d^r ij = |x^c i - x^c j| + |y^c i - y^c j| \quad \text{(equação 2)}$$

Distância retangular, e

$$d^E ij = \sqrt{(x^c i - x^c j)^2 + (y^c i - y^c j)^2} \quad \text{(equação 3)}$$

Distância euclidiana.

É importante destacar que para estimar as distâncias entre departamentos, no *layout* de departamentos, usa-se a distância de centro a centro, quando os pontos de entrada e saída são conhecidos.

Como mensionado anteriormente, o objetivo da busca tabu é minimizar o custo total de interconexão entre todos os departamentos, sejam fixos ou móveis, como apresentado matematicamente pela equação 4:

$$Min_{s \in S} \; F = \sum_i^n \sum_j^n C_{ij} F_{ij} D_{ij} M^{(k+l+m)} \quad \text{(equação 4)}$$

em que i, j = 1, 2, ..., n

sendo:

*Cij*: custo do fluxo entre o departamento i e o departamento j por unidade de distancia;

*Fij*: fluxo entre o departamento i e o departamento j;

*Dij*: distância retangular centro a centro entre o departamento i e j;

*k*: número de departamentos não posicionados;

*l*: número de departamentos que violaram adjacências (perto);

*m*: número de departamentos que violaram adjacências (longe);

*M*: valor positivo e alto definido em função do problema. Se restrições k, l e m forem violadas, a função de avaliação será penalizada;

*n*: porte do problema a ser tratado, ou seja, quantidade de variáveis a serem otimizadas;

As restrições de razão de aspecto são definidas como:

$a_{i(\min)} \leq a_i \leq a_{i(máx)}$ e $0 \leq \beta_i \leq \beta_{i(máx)}$ (equação 5),

em que:

*ai*: razão de aspecto do departamento i;

$ai_{(min)}$: razão de aspecto mínima do departamento i;

$ai_{(máx)}$: razão de aspecto máxima do departamento i;

$\beta i$: razão de área morta do departamento i;

$\beta i_{(max)}$: razão de área morta do departamento i;

O processo de resolução é iniciado com o fornecimento dos seguintes dados:

- Área total necessária (ATN): as dimensões do ambiente todo.
- Solução inicial viável (SIV): gera uma solução de partida factível.
- Algoritmo matriz tabu (AMT): gerencia a lista de elementos tabu.
- Algoritmo de busca tabu (ABT): busca a melhor solução na vizinhança atual.
- Algoritmo de alocação (VCM): executa a alocação física dos departamentos dentro do quadro externo.

A obtenção da solução inicial começa com o procedimento SIV construindo a lista de alocação que definirá, para o VCM, a seqüência de alocação dos departamentos fixos e móveis. Na lista, entra primeiro o departamento com maior área, seguido daquele com o qual tem melhor relacionamento. Em seguida é alocado o departamento com melhor relacionamento com o até então último departamento alocado e assim por diante. Se o último da lista não se relaciona com nenhum outro, o próximo departamento candidato a entrar na lista será o de maior área que ainda não se encontra na lista. Com a lista construída, passa-se ao algoritmo VCM.

O algoritmo VCM aloca primeiramente departamentos fixos e depois os móveis escolhendo as melhores posições e evitando violações de restrição.

O primeiro departamento é alocado no extremo esquerdo e superior do ambiente industrial. O próximo elemento da lista é posicionado à direita do departamento já alocado. Caso não seja possível, o VCM tenta a alocação do departamento, por deslizamento, abaixo. Caso ainda não seja possível, tenta-se à esquerda. Não conseguindo, tenta-se acima. Se mesmo assim não der, o departamento corrente pode ser atribuído a próxima localização livre dentro do *layout*. O procedimento continua até que todos os departamentos estejam alocados.

É no melhoramento da solução inicial que entra em operação a busca tabu ou como é também conhecido, a heurística tabu. A busca tabu explora a vizinhança de uma dada solução e seleciona a melhor, mesmo que esta piore a solução inicial. Os passos genéricos da otimização com base em busca tabu são:

- Partir de uma solução inicial D;
- Iniciar os parâmetros da lista tabu <- 0 e Melhor Solução <- D.
- Enquanto um critério de término não for satisfeito, avaliar a lista de candidatos a movimento (candidato = {D´ Є N(D), D´ ϶ a lista tabu ou D´ satisfaz a um critério de aspiração} e selecionar a melhor solução admissível; D* Є a soluções candidatas.
- Atualizar a lista tabu; atualizar a solução corrente D <- D*; se F(D) < F (Melhor solução), então, Melhor solução <- D;
- Retornar a melhor solução encontrada.

### 2.3.3 Algoritmos Genéticos

O algoritmo genético é uma técnica de procura estocástica. Ele pode explorar o espaço de soluções tomando o conceito da genética natural e teoria da evolução, tanto que usa muitos termos originados no campo da biologia. Em anos mais recentes, os AGs tem sido proposto como uma abordagem inovadora para a solução problemas de *layout* industrial. A AG começa com uma lista de soluções randômicas para o problema sob consideração. Esta solução é

chamada de população. Cada cromossomo da população recebe uma nota, denominada de aptidão. Os melhores cromossomos (mais aptos) são selecionados e os piores descartados. Os selecionados podem sofrer cruzamento e mutação, gerando descendentes para a próxima geração. O processo continua até que a solução satisfatória seja alcançada (ARGOUD, 2007). Na Figura 5 é ilustrado os componentes do algoritmo genético.

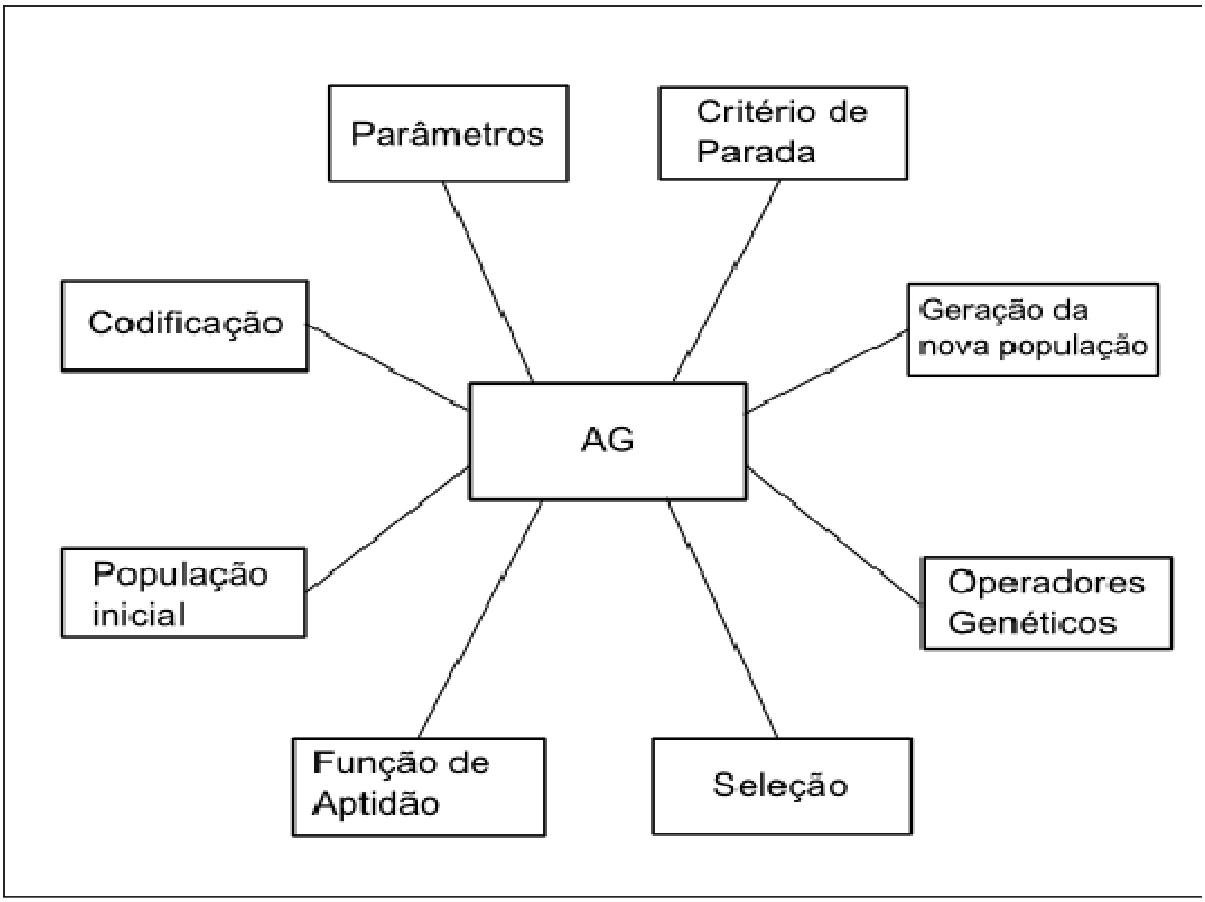


**Figura 4: Componentes do algoritmo genético.**
**Fonte: Argoud (2007).**

A codificação é a forma com o qual cada cromossomo irá representar a solução. Por exemplo, o cromossomo (3 3 3 1 3 2 2 1) pode representar o conjunto de todas as máquinas do *layout* enquanto que cada valor do gene pode significar a célula ao qual a máquina pertence. Porém Argoud (2007) usou uma codificação chamada de grupo-número, onde, por exemplo, o cromossomo (2 1 2 1 3 | 2 2 3 3 1) representa respectivamente, as máquinas e as peças, os valores dos genes representam a célula ao qual pertencem.

A geração da população inicial deve conter os cromossomos representativos do espaço de soluções. A forma mais comum de geração da população inicial é a aleatória. Outra forma é gerar a primeira metade da população de modo aleatório e a outra metade por troca de bits.

Em um algoritmo genético, a função de aptidão é atribuir nota a cada cromossomo e assim avaliar a qualidade da evolução. Também é usada para selecionar cromossomos para reprodução.

A função do operador genético é fazer com que a população inicial sofra transformações e o AG melhore com o tempo. Os operadores de cruzamento e mutação são os principais mecanismos de busca em regiões desconhecidas do espaço de soluções.

O operador de mutação introduz novo material genético na população, de modo a fazer o AG partir para buscas em regiões ainda não exploradas do espaço de busca. A mutação é realizada após o cruzamento e aplicada de acordo com a taxa de probabilidade e consiste em alterar o valor de um ou mais genes do cromossomo.

O processo de seleção refere-se a escolha dos indivíduos pais que irão gerar os descendentes através dos operadores de cruzamento e mutação. Os métodos comumente usados são por roleta, por ranking e por torneio (ARGOUD, 2007).

## 2.4 Os Softwares de Modelagem Tridimensional

Para o desenvolvimento de arranjos físico seguindo a proposta de que o processo de projeto deva ser visualmente mais próximo da realidade a ser estudada, com melhor ergonomia e economia de tempo e custo, resolveu-se dispor de softwares de modelagem tridimensional voltado a aplicações artísticas.

Duas opções foram consideradas. O Blender 3D, produzido pela comunidade Blender e o Maya, fornecido pela empresa Autodesk. As duas opções de software têm como seu objetivo central, a modelagem tridimensional voltado para produções artísticas.

A atenção então adotada para esse tipo de software foi justamente a facilidade da qual máquina e equipamento pode ser modelado de modo que se tornar modelo de representação idêntico ao objeto real.

Outra questão de grande interesse está na linguagem de *scripts*. É a interface de edição de *scripts* que permitir buscar uma customização desse *software* para a aplicação de modelagem de arranjos físico.

### 2.4.1 Maya

O Autodesk Maya é uma solução integrada para modelagem 3D, animação, efeitos especiais e renderização. Ele tem uma arquitetura aberta, ou seja, todo o trabalho pode ser codificado ou programado utilizando uma API (*Aplication Programming Interface*) ou as linguagens de *scripts*, como o MEL (*Maya Embedded Language*).

O Maya é uma combinação original de três linhas de *software*: Wavefront, Thomson Digital Image (TDI) Explore e Alias Power Animator. Em 1993, Wavefront adquiriu a TDI e em 1995 a Silicon Graphics Incorporeted (SGI) adquiriu ambas.

Wavefront e Alias estavam trabalhando em um projeto de *software* conjunto. Codinome Maya. Maya foi desenvolvido em próxima colaboração com o estúdio Walt Disney, e portanto, a interface gráfica do usuário foi customizada para atender os requisitos de uma empresa que tinha décadas de experiência em animação.

Em 2003, a Alias foi vendida a SGI e em 2005, novamente vendida a empresa Autodesk.

### 2.4.2 Blender

Segundo Brito (2007), o Blender é um *software* de modelagem e animação 3D de código aberto. Ele está sob a licença GNU-GPL, que permite a qualquer pessoa ter acesso ao código-fonte do programa para que possa fazer melhorias, contanto que disponibilize tais melhorias à

comunidade. Além de fazer modelagem e animação, o Blender permite-nos criar jogos sem o uso de programação e realizar pós produção de animações com um editor de vídeo integrado.

O Blender começa com a decisão do estúdio de animação holandês NeoGeo em desenvolver ferramenta de modelagem e animação para uso interno. Em 1998 uma empresa derivada da NeoGeo foi criada para desenvolver esse *software* e prover serviços baseados nele. Not a Number era o nome, conhecida pela sigla NaN.

Em 2001, o primeiro *software* comercial da NaN foi lançado, o Blender Publisher, direcionado ao mercado de internet interativa 3D. Porém o fraco desempenho comercial do *software* e dificuldades de mercado fizeram com que a NaN encerrasse suas atividades e o desenvolvimento do *software* foi interrompido.

Em março de 2002, foi criada a Fundação Blender, sem fins lucrativos, com o objetivo de dar continuidade ao desenvolvimento do *software* como projeto de código aberto com o auxílio da comunidade de usuários que já conhecia o *software*.

No mesmo ano a campanha “Blender livre” arrecadou doações para que a fundação pudesse adquirir os direitos sobre o código-fonte e a propriedade intelectual e conseguiu lançar o Blender como projeto de código aberto.

O Blender também tem uma interface para inserção e programação de *scripts* baseada na linguagem *Python*, também de código aberto.

# 3 METODOLOGIA

A presente pesquisa é de natureza aplicada com caráter experimental.

O projeto irá avaliar a possibilidade de alguns procedimentos de alocação de máquinas no *layout* visto na literatura de Administração da Produção tornar aplicação em *software* de modelagem tridimensional. Caso esses procedimentos sejam transformáveis em algoritmos computacionais, o projeto objetiva propor uma aplicação prática usando um *software* de modelagem tridimensional. Caso não o sejam, parte-se para a busca de algoritmos que faça essa proposta e que esteja próximo à linguagem de máquina, avalia-se os algoritmos quanto a sua aplicabilidade e faz-se a aplicação.

Inicialmente, foi efetuado pesquisa na literatura de Administração da Produção sobre os algoritmos de ordenação de máquinas e estações de trabalho no *layout* industrial. Nessa pesquisa, foram escolhidos três algoritmos, o que não quer dizer que outros sejam ou não sejam aplicáveis em *software*.

O passo seguinte foi o levantamento das necessidades que um aplicativo desse tipo requer. Foi confrontado as necessidades dos possíveis usuários da aplicação com as necessidades do sistema escolhido entre as duas opções.

Passou então para o trabalho de descobrir e modelar o modo como os algoritmos irão ser alocados dentro da aplicação. Como será o acesso aos algoritmos, em que menus e submenus eles estarão residentes. Como acessar os algoritmos.

Foi efetuado o trabalho de implementação dos algoritmos, entre eles os de ordenação e também os de auxílio ao design. E por último, é discutido os resultados, limitação, dificuldades e propostas futuras.

# 4 O DESENVOLVIMETNO DA APLICAÇÃO

A proposta de desenvolvimento de aplicação voltada ao auxílio no design de *layout* de instalações industriais irá se basear no conceito de reuso da engenharia de *software*. A proposta está no desenvolvimento de *scripts* e componentes que estarão residentes no *software* de modelagem e animação tridimensional como o Blender 3D ou o Autodesk Maya. Das duas opções de *software*, foi considerado na pesquisa o Blender 3D devido às seguintes características:

a) Código Aberto: o *software* está disponível gratuitamente no site www.blender.org sob a licença GNU-GPL (*Gnu General Public License*);
b) Menor tamanho total da aplicação: são 50 MB de espaço requerido após a sua instalação;
c) Maior quantidade de *scripts* a disposição nos sites das comunidades de usuários do *software*;
d) Gratuito.

## 4.1 Desenvolvimento de *Software* Orientado a Objetos

Nesta seção coloca-se a questão do paradigma de orientação a objetos e construção de componentes para contextualizar a modelagem das necessidades da aplicação e o modo de desenvolvimento das ferramentas.

### 4.1.1 Considerações iniciais

Desde os anos 80, o conceito de orientação a objetos (OO) vem sendo aplicados a desenvolvimento de *software*. Nos anos 90, a OO ganhou força devido a grande volume de pesquisa na sua utilização e aplicação e interesse das empresas de desenvolvimento de *software* em suas vantagens e benefícios, tornando-se largamente utilizado hoje em dia.

Seus conceitos facilitam o desenvolvimento em todas as fases do processo e possibilitam mudanças rápidas e conduzem ao reuso. Atenção deve ser dada ao reuso, pois na proposta é a características mais importantes da OO a ser explorada.

Além disso, sistemas orientados a objeto são mais fáceis de serem mantidos, de adaptar e de ampliar devido a estrutura desacoplada. Mesmo propondo um novo paradigma para o desenvolvimento de *software*, a popularidade da OO vem crescendo continuamente e a grande maioria dos sistemas atuais são orientados a objetos.

### 4.1.2 Princípios de orientação a objetos

A grande diferença entre uma abordagem convencional ou estruturada e uma orientada a objetos está na forma de visualização dos dados e processos que compõe o *software*. Na abordagem convencional o *software* é visto como uma coleção de processos dentro do sistema, enquanto na abordagem orientada a objeto, essas rotinas e dados são encapsulados em uma única entidade chamada objeto.

Um objeto pode ser visto como uma entidade que possui um estado, atributos e um conjunto de operações ou métodos que alteram seu estado. Cada operação associada ao objeto fornece serviços a outros objetos que o solicitam quando algum processo computacional é requisitado.Os objetos são criados de acordo com as definições de uma classe de objetos que serve de modelo para a criação dos mesmos. Isso inclui a declaração dos atributos e operações que devem ser associados ao objeto dessa classe (SOMMERVILLE, 2001).

Um *software* orientado a objetos é formado por um conjunto finito de objetos que se comunicação entre si através de mensagens. Cada mensagem é um sinal enviado de um objeto a outro solicitando um serviço.

A definição acima mostra visão diferenciada. No entanto, o grande diferencial está relacionado a três características fundamentais da OO conhecidas como encapsulamento, herança e polimorfismo.

- O encapsulamento é inerente à definição de objeto e fornece diversos benefícios como: detalhes internos da implementação das operações são ocultados do mundo externo; as operações e dados são reunidas em uma única entidade (isso facilita o reuso dos componentes) e as interfaces entre os objetos são simplificadas;

- A herança é o mecanismo que possibilita a um objeto específico incorporar atributos, operações e relacionamentos de objetos genéricos sem repetir o código, e definir novos atributos, operações e relacionamentos entre si. Isso é feito em nível de classes, onde classes especializadas (subclasses) compartilhem as mesmas características, isto é, tenham o mesmo ancestral (superclasse). É um diferenciador chave, afinal possibilitam forte encapsulamento, reutilização de código e fácil manutenção;

- O polimorfismo é o mecanismo que possibilita a uma operação assumir vários comportamentos diferentes, dependendo do contexto e sobre o que opera.

### 4.1.3 O conceito de reuso

Segundo Pressman (2006):

> O reuso, no contexto da engenharia de *software*, não é uma idéia velha nem nova. Os programadores têm usado idéias, abstrações e processos desde os primeiros dias da computação, mas as primeiras abordagens ao reuso eram as da prática atual. Hoje, sistemas complexos e de alta qualidade baseados em computador precisam ser construídos em períodos de tempo muito curtos e demandam uma abordagem mais organizada de reuso...

O conceito de reuso está na literatura (PRESSMAN, 2006) abordada sob o desenvolvimento baseado em componentes (CBSE – *Component-Based Software Engineering*).

Superficialmente, a CBSE parece bastante semelhante à engenharia de *software* convencional ou orientada a objetos. Uma equipe de *software* estabelece os requisitos para um sistema a ser construído usando técnicas de elicitação de requisitos, partindo então para um projeto arquitetural. Daí porém, ao invés de partir para tarefas mais detalhadas, a equipe examina os requisitos para determinar que subconjunto é mais adequado à composição do que à construção.

O modelo de análise é avaliado para determinar os elementos do modelo que indicam componentes reusáveis existentes. O problema é extrair informação do modelo de requisitos com o objetivo de conseguir “satisfazer a especificação”. Se satisfizer a especificação leva a

componentes que atendem às necessidades, o projetista de *software* pode extrair esses componentes de uma biblioteca de reuso (um repositório) e usá-los no projeto dos novos sistemas. Caso esses componentes não puderem ser encontrados, o engenheiro deve aplicar métodos convencionais ou OO para criá-los (os componentes). É nesse ponto – quando o engenheiro de *software* começa a criar um novo componente – que o projeto para reuso (*design for reuse*, DFR) deve ser considerado.

Todo o conceito exposto acima serve de base para a proposta de se reutilizar um *software* inteiro, no caso o Blender 3D, da qual, ainda que voltado para modelagem e animação artística, como um *software* de modelagem voltado ao design de *layout* industrial, através de componentes construídos e então embutidos no próprio.

## 4.2 O Levantamento do Problema do Design de Arranjos Físicos

O levantamento dos requisitos permite revelar o domínio do problema enquanto sua análise mostra o domínio da aplicação e direciona para a solução computacional mais adequada.
No Quadro 1, é mostrado o levantamento das necessidades de um usuário, confrontado com os requisitos do sistema:

| **Necessidades do sistema** | **Requisitos do sistema** |
|---|---|
| Visualização da planta industrial em perspectiva tridimensional | O *software* deve ser do tipo especialista voltado à modelagem de objetos tridimensionais. |
| Capacidade para permitir inclusão de novos algoritmos e operações auxiliares para atender as necessidades futuras ou não previstas dos projetistas. | Além de flexível, o *software* deve ter estrutura preparada para a inclusão de novos algoritmos ou *scripts* sem sofrer modificações em seu código fonte. |
| Organização dos algoritmos e *scripts* de acordo com as necessidades de cada método ou projeto para a facilitação da escolha. | O *software* proposto deve criar um ambiente de trabalho em que os algoritmos sejam organizados de acordo com cada método, projeto ou tipo de arranjo físico. |
| Deve conter uma biblioteca pronta de modelos tridimensionais de máquinas e estações de trabalho, de modo a libertar o projetista da árdua tarefa de modelagem tridimensional. | O sistema proposto deve conter uma boa gama de máquinas e estações de trabalho disponíveis e de simples inserção dentro da cena 3D, sendo essas máquinas de vários tipos de indústrias. |

**Quadro 1 – Levantamento e Análise de Requisitos. Adaptado de Tiberti (2003).**

*Software*s desses domínios possuem características estruturais bem definidas. Em geral, possuem: (1) um conjunto de rotinas que implementa funções de cálculo e análise; (2) um conjunto de mecanismos para seleção, controle e execução dessas funções a partir de critérios e funções bem definidas; (3) um conjunto de interfaces para a interação do usuário com o *software* e com as rotinas, mecanismos e operações mencionadas.

No *software* proposto essas características foram complementadas com base no quadro. A estrutura arquitetural básica observada nesses *software*s é composta por: (1) pacotes ou biblioteca de rotinas; (2) um módulo que cuida da interação com o usuário; e (3) algum tipo de repositório de dados.

Durante o processo de análise, foram identificados quatro elementos essenciais para o uso do *software* proposto, denominado elementos da aplicação:

a) **dados do projeto:** quaisquer dados obtidos para a aplicação dos diversos algoritmos e métodos usados nos processos de análise e formação de arranjo físico de um projeto, assim como quaisquer resultados;
b) **algoritmos:** quaisquer algoritmos aplicados nos processos de análise e formação de arranjos físicos;
c) **operações auxiliares:** quaisquer rotinas usadas para auxílio na visualização, medição e avaliação e comparação dos resultados;
d) **interfaces com o usuário:** quaisquer interfaces do *software* usadas para a interação do projetista com os algoritmos e operações auxiliares ou com outras funções definidas para o *software*.

Esses elementos compõem a base para as aplicações do *software* proposto como *software* de apoio ao projeto de arranjo físico.

Outro elemento importante são as estruturas de projeto. Estruturas de projeto são estruturas dinâmicas usadas para organizar os elementos da aplicação de acordo com as necessidades do projeto, métodos ou tipos de arranjos físico.

## 4.3 Estruturas de Projeto

As estruturas de projeto foram elaboradas como forma de encaixar os componentes ou *scripts* na aplicação Blender 3D.

Uma estrutura de projeto é descrita como uma forma de organização estrutural hierárquica que irá relacionar um projeto com os componentes que descrevem seus dados. Esses componentes estão relacionados com os componentes que implementam algoritmos e operações auxiliares que podem ser usados para processar esse projeto. Um projeto pode ser dividido em vários subprojetos. Um subprojeto é uma denominação genérica para representar um método, um arranjo físico ou até outro projeto e é composto da mesma maneira, podendo ainda ser dividido em outros projetos. Essa forma genérica irá permitir representar desde um método a um projeto complexo, dividido em vários tipos de arranjos físico com suas diversas visões gerais e detalhadas. A Figura 6 mostra um exemplo de estrutura de projeto para um arranjo físico.

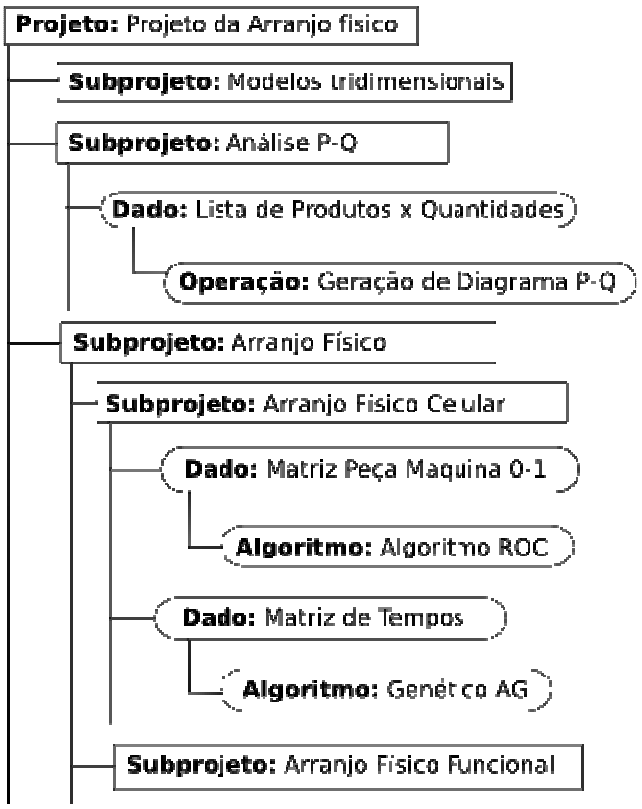


**Figura 6 – Estrutura de Projeto de um dos Arranjos Físicos.**
**Fonte: Tiberti (2003)**

No Blender 3D a estrutura de projeto muitas vezes irá se confundir com a aplicação Blender. Isso porque o desenvolvimento irá buscar na *Blender-Python* API os métodos e operações de transformação geométrica nativas no próprio *software*. Por isso ao contrário da proposta de *framework* para construção de aplicativos de apoio ao design de arranjos físicos apresentado por Tiberti (2003) que tem em seu desenvolvimento a chamada Estrutura de Aplicação, aqui não o haverá tal estrutura. A estrutura de interface do Blender para acesso ao módulo responsável pelo auxílio ao design de arranjos físicos industriais pode ser representado da seguinte maneira na Figura 7.

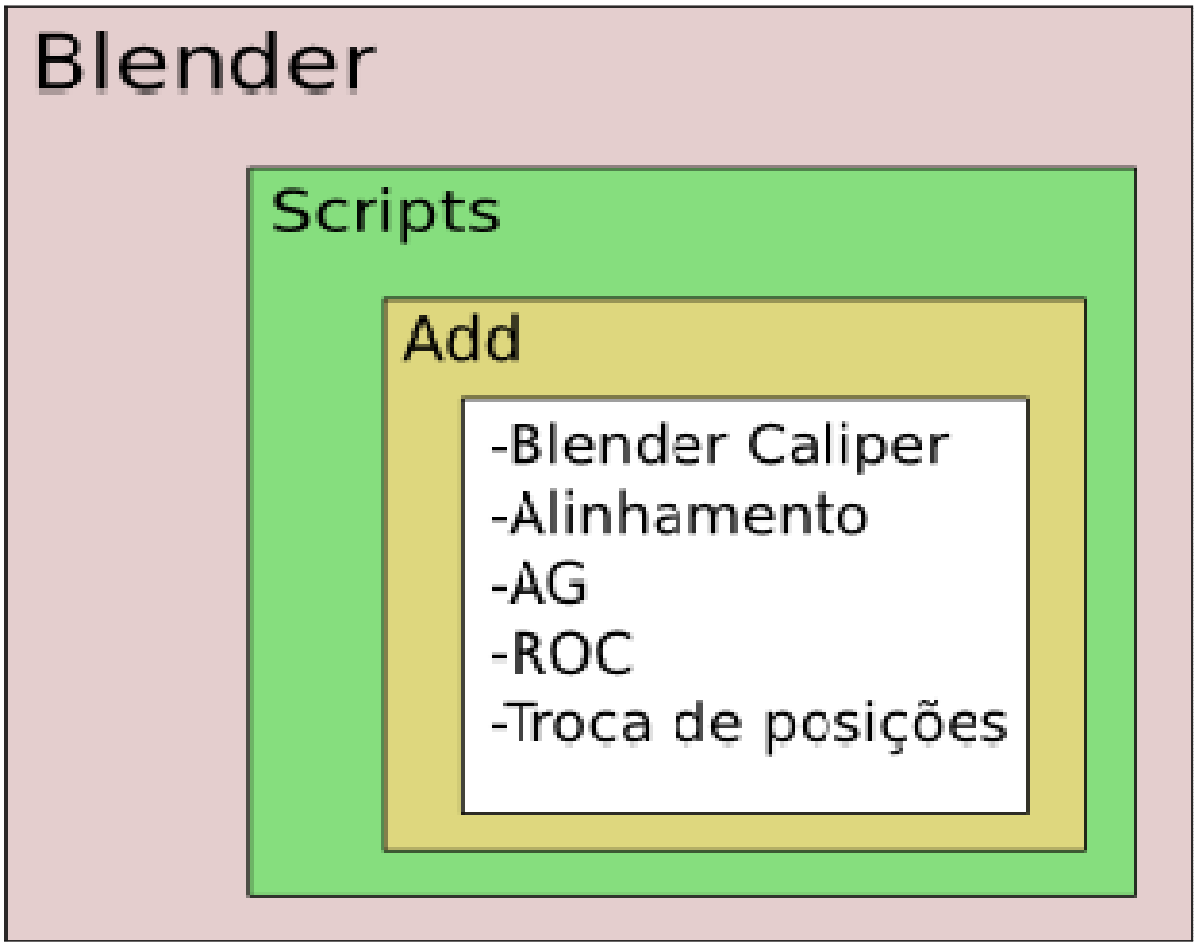


**Figura 7 – Estrutura da Aplicação Blender 3D**

E pode ser conferido na Figura 8 como ficará o acesso ao módulo que dará auxílio ao *design* de arranjos físico.

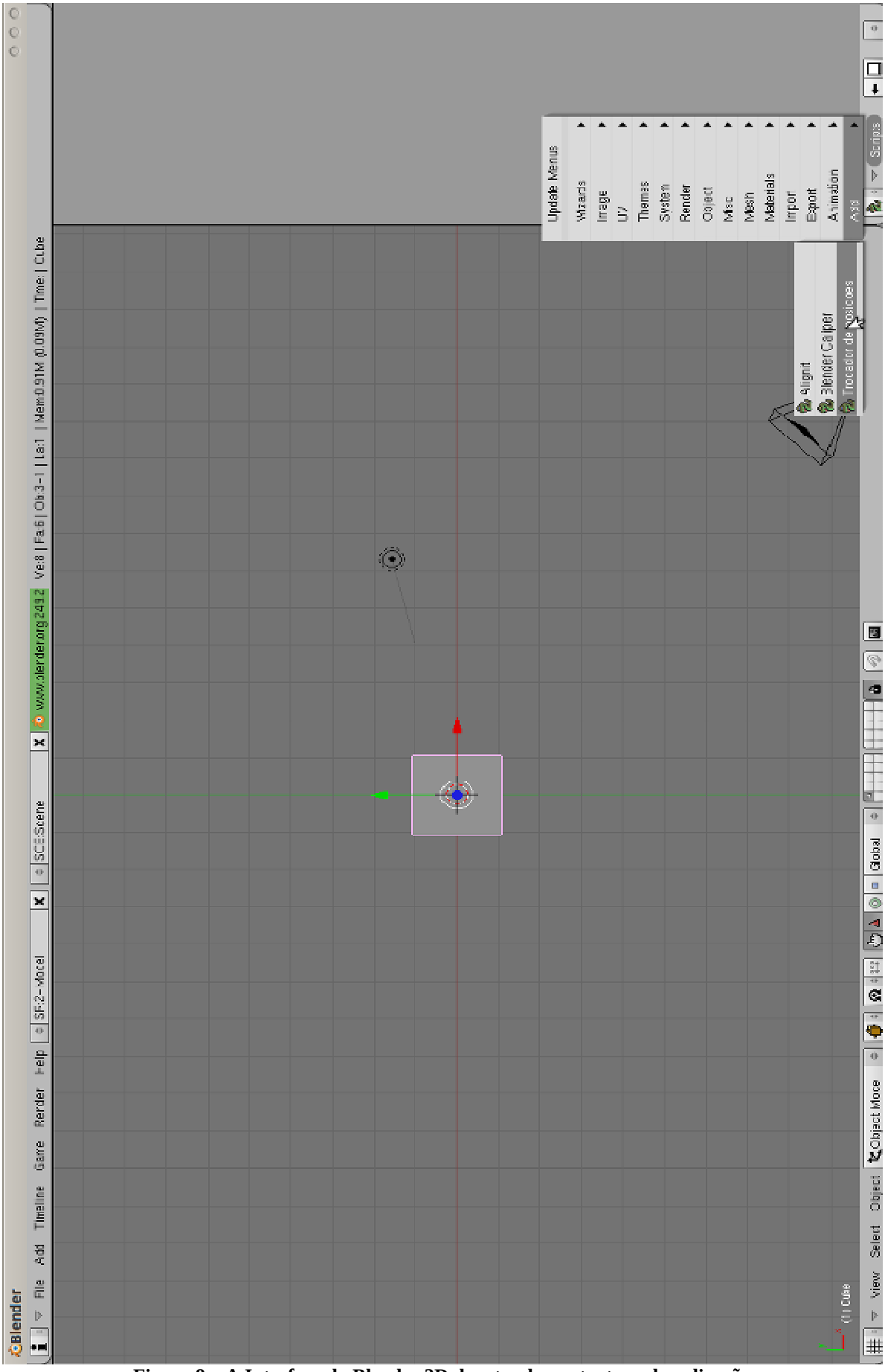


**Figura 8 – A Interface do Blender 3D denotando a estrutura de aplicação**

A Figura 7 representa o conceito de encapsulamento das ferramentas na aplicação Blender 3D. As ferramentas ‘Blender Caliper’, ‘Alinhamento’, ‘AG’, ‘ROC’ e ‘Troca de Posições’ estão inseridas no menu ‘Add’. Este, por sua vez, está inserido no menu ‘*Scripts*’, menu este onde ficam armazenados todos os *scripts* já incorporados ao Blender nativamente ou inseridos pelo usuário ou desenvolvedor. E, por fim, o menu ‘*Scripts*’ é um componente, ou seja, faz parte do todo que é a aplicação.

A Figura 8 denota na prática a localização das ferramentas inseridas. Para se acessar a ferramenta ‘Troca de posições’, estando com a aplicação inicializada, é necessário acessar o menu ‘Scripts’ que está na barra lateral do programa. Após isso, acessar o sub-menu ‘Add’, finalmente, o ‘Trocador de posições’.

E por último, para reforçar o caráter mais intrínseco do projeto de *layout* industrial dentro do Blender, é importante salientar que o modelo tridimensional de máquinas e equipamentos são construídos e sofrem as transformações geométricas dentro do Blender como se fosse um trabalho de design convencional.

# 5 O DESENVOLVIMENTO DAS FERRAMENTAS EM *PYTHON*

*Python* é uma linguagem largamente usada, de propósito geral e orientado a objetos. É usado para tudo, de pesquisa até programação web e tem uma variedade de bibliotecas terceiras e ferramentas disponíveis para ele. A linguagem *python* é usada para criação de *scripts* no Blender, e até uma última informação, também no Maya da Autodesk e no Cinema 4D da Maxon.

Usando o interpretador *Python* embutido no Blender e a API para rodar os *scripts*, é possível acessar os recursos do Blender e funcionalidades. Isto é útil para um larga variedade de tarefas de simples automação porém, repetitiva que seria tedioso fazer manualmente através da interface padrão. É útil para confeccionar as próprias macros e ferramentas e criar sofisticadas visualizações matemáticas automaticamente ao clique de uma tecla.

As facilidades de uso do *Python* não se estendem somente a aprendizagem, mas também ao desenvolvimento. O código é conhecido por sua legibilidade. Uma das mais notáveis peculiaridades do *Python*, sua sintáxe, é também um fator de grande importância na escrita de códigos legíveis. Em *Python*, o escopo de controle de estruturas tais como o laço **for** e condicionais é determinado por identação da linha.

Em adição ao alto nível de abstração para simplificar tarefas e tomar vantagens em legibilidade apoiado pela sintaxe, o *Python* ganhou status de uma verdadeira linguagem de programação orientado a objetos (OOP) tomando vantagem do paradigma da OOP e os benefícios que o paradigma trás. Como explanado acima, a OOP habilita o programador a definir as estruturas de dados que serão usadas em seus programas, que são as classes. As instâncias de uma classe são chamadas de objetos desta classe. Muito da OOP envolve envio e recepção de consultas, chamados de métodos, entre objetos. Cada classe tem específicos métodos definidos para ele.

A vantagem da OOP é que após a compreensão do básico da linguagem de programação, interagir com as bibliotecas que foram escritos por outras pessoas exige apenas que se saiba que com quais classes está lidando e quais os métodos que devem usar com essas classes para obter os resultados desejados. A referência completa de classes, métodos e variáveis especiais de um determinado *framework* é chamado de *application programming interface* (API). É

nesse momento que é necessário entrar um pouco na Blender API, que será o recurso de informação nos acessos as classes necessárias para confeccionar os *scripts* no Blender.

## 5.1 O *Blender-Python* API

O mais importante documento para a construção de *scripts* em Blender é API. Lá contém uma completa lista de todos os módulos definidos para o Blender, todos os módulos de classes, atributos e métodos definidos para os objetos dessas classes. Se há necessidade de escrever *scripts* para o Blender, a API é a ferramenta de informação a ser utilizada. A API do Blender versão 2.49b pode ser encontrado on-line em www.blender.org/documentation/249PythonDoc/.

Como exemplo do uso da API, poder ser visto na Figura 9, os passos e comando para se criar no Blender 3D um cubo com a cor roxa.

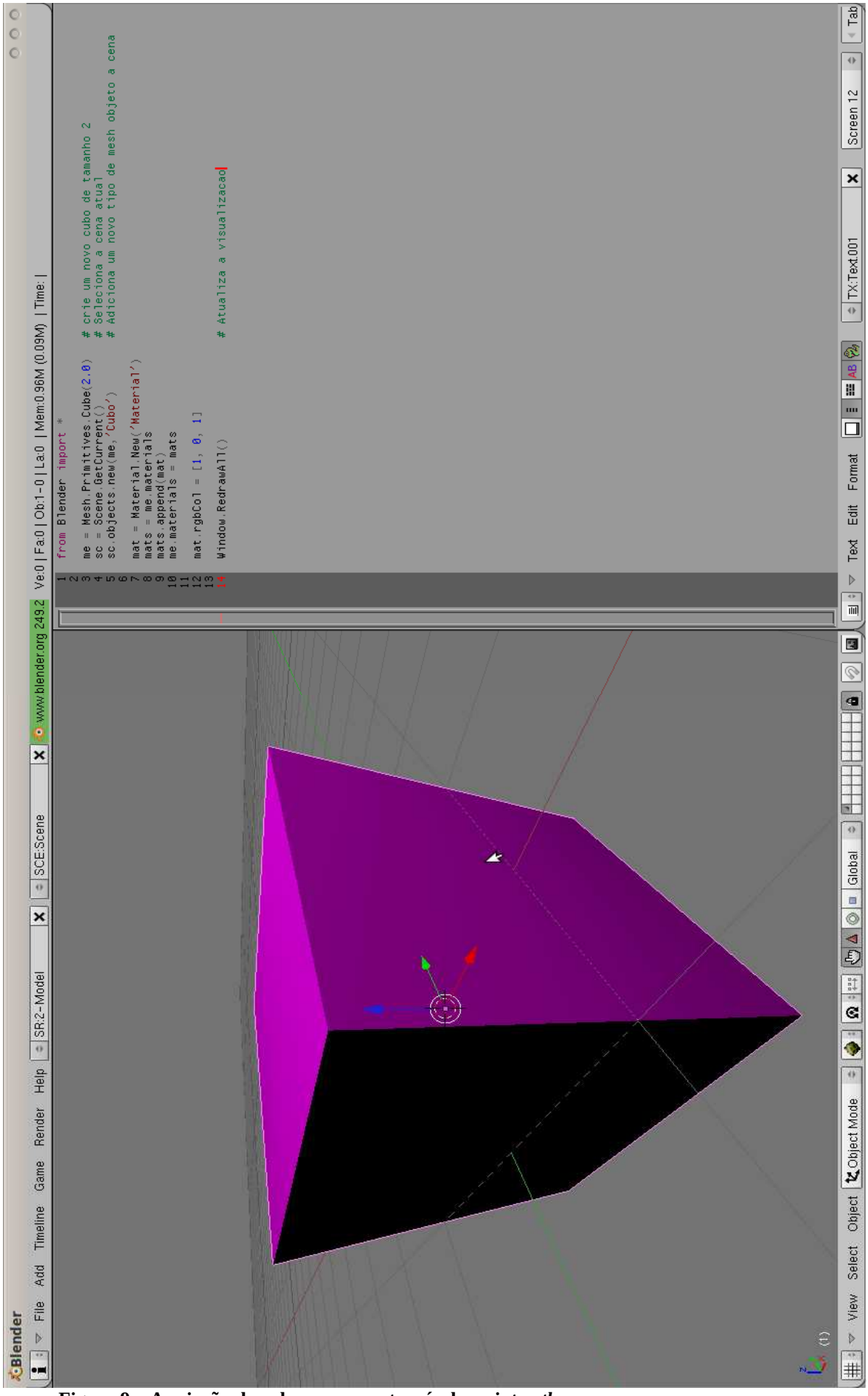


**Figura 9 – A criação do cubo em roxo através de script *python***

Na Figura 10, é mostrado com mais detalhes, o *script* para a criação do cubo com a cor roxa. Importante a partir de agora, é fazer uma convenção para distinguir o objeto tridimensional no *software* com o objeto sendo a instância de classe do paradigma OO. Para se fazer referência ao objeto 3D, fala-se do modelo tridimensional, enquanto a palavra objeto ainda irá fazer referência à instância de alguma classe de algum módulo do Blender.

```
import Blender
from Blender import *

me = Mesh.Primitives.Cube(2.0)  # crie um novo cubo de tamanho 2
sc = Scene.GetCurrent()             # Seleciona a cena atual
sc.objects.new(me,'Cubo')       # Adiciona um novo tipo de mesh objeto a cena

mat = Material.New('Material')  # cria um um novo material
mats = me.materials  #
mats.append(mat)
me.materials = mats

mat.rgbCol = [1, 0, 1]

Window.RedrawAll()              # Atualiza a visualizacao
```

**Figura 10 – O script *python* da criação do cubo em detalhe**

A primeira linha é a importação para o script do principal módulo que é o módulo *Blender*. Consultando a *Blender-Python* API, tem-se dois módulos principais que abarcam todos os submódulos com suas classes e métodos, que são os módulos *Blender* e *Bpy*. Internamente esses dois módulos estão contidos seus submódulos. Como exemplos desses submódulos podemos listar: *Mesh*, *Armature*, *Camera* e etc.

Na segunda linha, o * indica que do módulo Blender estamos importando todos os submódulos, da quais iremos usar: *Mesh.Primitives*, *Scene*, *Object* e *Material*. Na quarta linha, cria-se um objeto (não confundir com modelo tridimensional) com a declaração de uma função ou método do submódulo *Mesh.Primitives*, que é a função *Cube()*, função está que na definição da API *Blender-Python* é um método incorporado ao submódulo *Mesh.Primitives*. Se consultarmos a API, a mesma irá indicar o que a função *Cube()* irá fazer e qual tipo de parâmetro a função espera receber, no caso, um valor de ponto flutuante. Na quinta linha é

criada o objeto “sc” que é a instanciação da cena onde os modelos tridimensional estarão presentes. A sexta linha é efetivamente a colocação do modelo tridimensional (a instância *me*) com o nome 'Cubo'. Tanto na quinta quanto na sexta linha já estamos usando o submódulo *Scene*.

Nas cinco linhas posteriores estão os comandos *scripts* para a alocação da cor roxa no nosso cubo recém criado Há que se dizer algo sobre o sistema de Materiais. Criamos um objeto com a classe *Material*, da qual é definido no módulo Material na API. Para criar o novo material, usa-se o método *New()* para passar o novo objeto resultante para a variável **mat**. Tem-se agora um objeto Material chamado mat. O próximo passo é adicionar ele à coleção de materiais do objeto me. Neste ponto ocorre uma estranha e flagrante inconsistência na implementação da API. A coleção de materiais do *mesh* deveria se comportar como uma lista do *python*. Entretanto, a implementação corrente tem algumas inconsistências (para não dizer *bugs*), e esta é uma dessas inconsistências. De fato, não podemos simplesmente anexar **mat** ao mesh **me.materials** usando o método *append()*. Existe uma grotesca, porém, solução funcional para isto: Pode-se copiar me.materials para uma nova variável, que no caso foi chamada de **mats**, e então passar os valores de **mats** para o **me.materials**. Daí as três próximas linhas:

```
mats = me.materials
mats = append(mat)
me.materials = mats
```

O último comando *Window.RedrawAll()* é para atualizar a 3D View.

Para inserir o *script* no Blender, deve-se seguir certas convenções. O *script* descrito acima como foi demonstrado, funciona. Porém, para ele ser utilizável no modo como foi construído, ele deve ser adicionado como texto no Blender. Para tornar seu script parte do *software* incluso no menu ‘*Scripts*’, ele precisa seguir uma convenção, como pode ser visto na Figura 11.

```
#! BPY
"""
Name: 'Meu Script'
Blender: 249
Group: 'Mesh'
Tooltip: 'Irá criar um cubo de arestas no tamanho 2 na cor roxa
"""
```

**Figura 11 – Procedimento a ser inserido nos *scripts*.**

Todo *script*, para ser interpretado e incorporado pelo *software* Blender, precisa desse cabeçalho, que está mostrado na Figura 11. Ele é alocado logo no começo do arquivo. Se um *script* com esse cabeçalho é alocado na pasta de subdiretório .blender/*scripts* em seu diretório de instalação do Blender quando o mesmo for iniciado, o *script* construído irá aparecer no apropriado submenu do menu '*Scripts*' baseado no que está definido em *Group:*. No exemplo da Figura 11, o script denominado "Meu Script" irá aparecer no menu '*Scripts*' e no submenu *Mesh*.

No contexto do desenvolvimento dos componentes e *scripts* que irão dar suporte ao design de arranjos físicos, os métodos mais importantes são os que lidam com variáveis de instância LocX, LocY, RotX e RotY. Essas variáveis de instância, pertencentes a classe **Object** guardam em si, as localizações e posição de rotação respectivamente. Elas são importantes porque basicamente o trabalho com o design de *layout* industrial será de posicionamento e deslocamento dos modelos tridimensionais das máquinas e estações de trabalho através da planta e esta inserida numa "cena" do Blender.

É necessário salientar outra característica da aplicação que foi elencada nas necessidades do sistema e requisitos do sistema (ver Quadro 1). Trata da quarta necessidade do sistema, qual seja, de existir bibliotecas de máquinas e estações de trabalho. É particularmente importante essa necessidade, devido ao objetivo de se retirar do engenheiro ou projetista esse esforço de trabalho artístico e trazer esse esforço para o desenvolvedor da aplicação. Por isso, a Figura 11, mostra uma estação de trabalho já confeccionada e a disposição no aplicativo para ser inserida no *layout* em desenvolvimento sem maiores esforços.

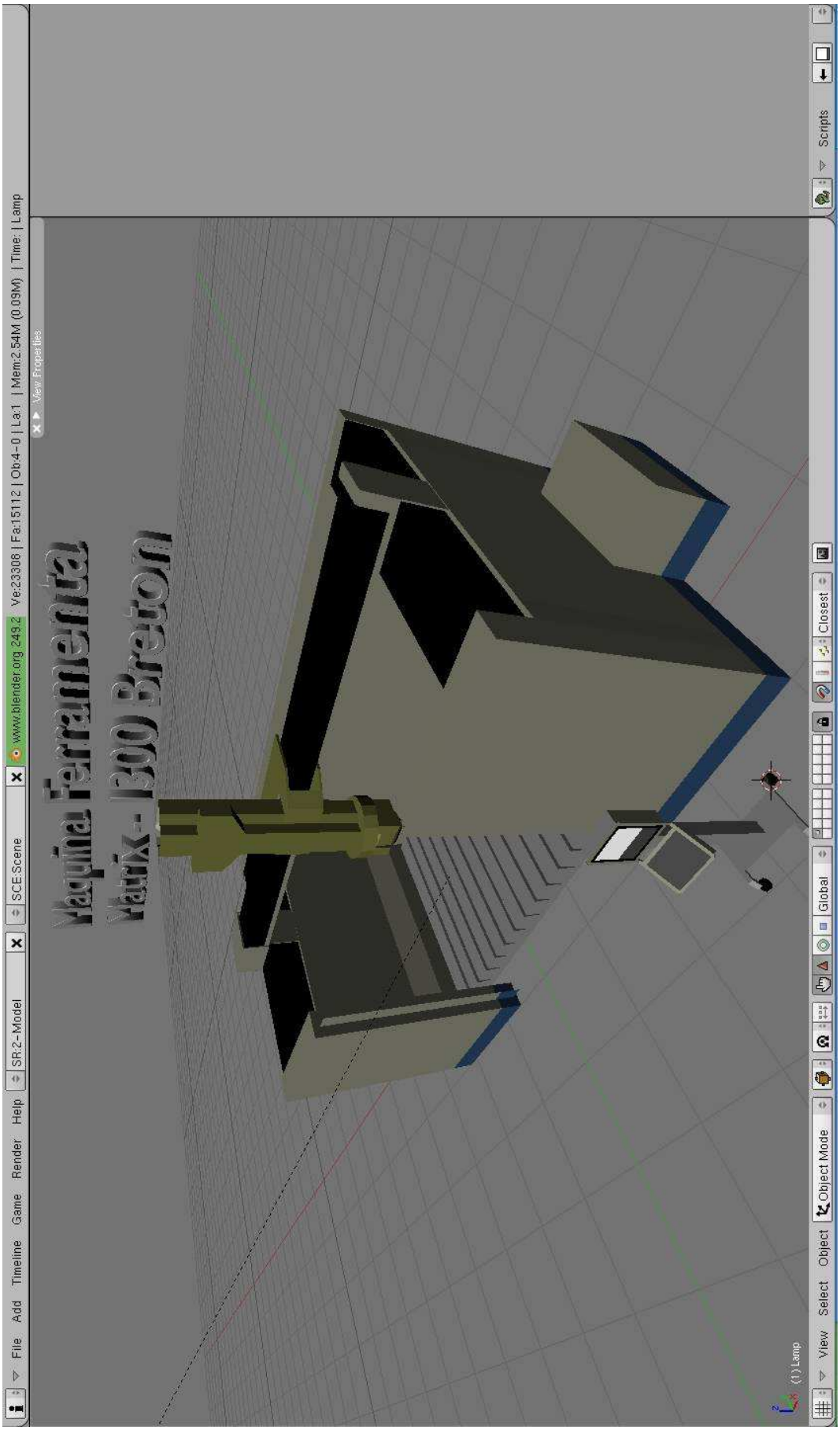


**Figura 12 – Modelo 3D de máquina importada da biblioteca da aplicação**

Com isso, o trabalho de fazer o posicionamento do modelo tridimensional fica ergonomicamente melhor, pois o esforço artístico já foi demandado. Para incorporar o modelo tridimensional é necessário apenas uma série de comandos a serem executados na própria aplicação. Consiste é acionar o submenu "*Append or Link*" no menu "*File*", como está mostrado na Figura 13.

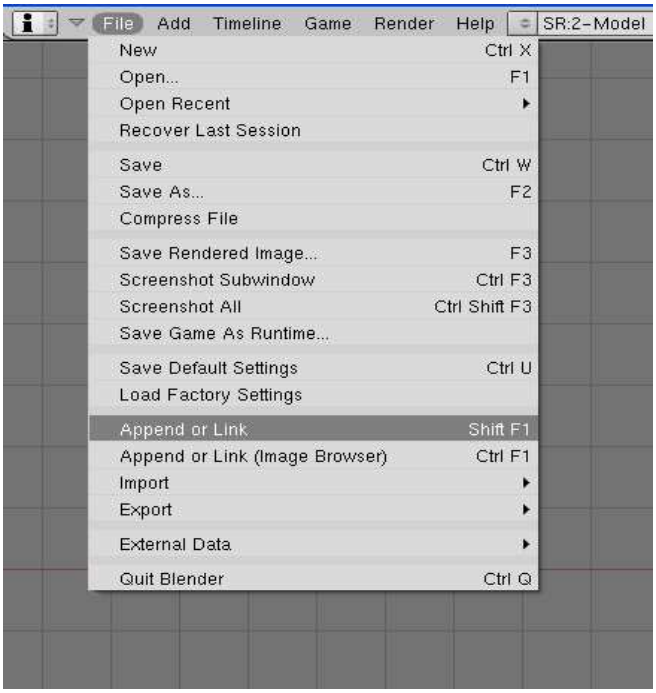


**Figura 13 – Acionamento do menu "Append or Link"**

Após isso, é só procurar pelo nome do seu arquivo correspondente ao modelo 3D que se deseja incorporar ao *layout* de fábrica. Essa ação é mostrada na Figuras 14 e 15.

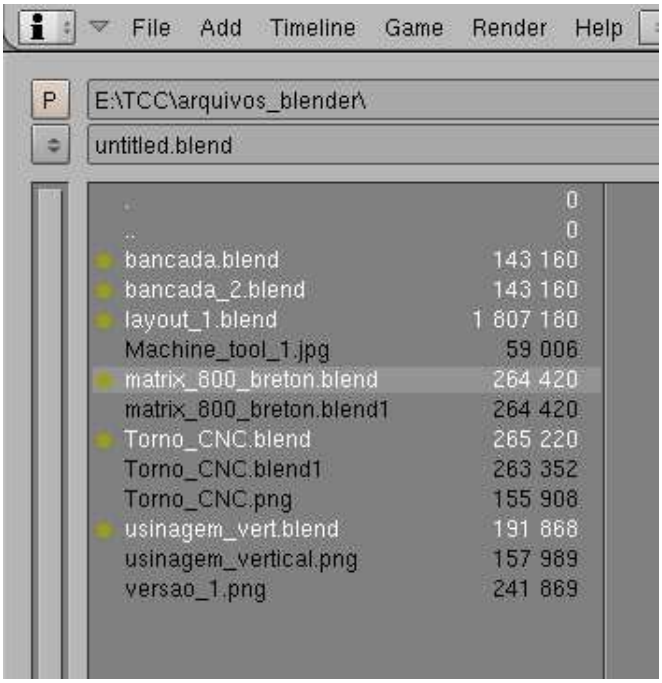


**Figura 14 – Seleção do modelo 3D.**


**Figura 15 – O Botão “Load Library” carrega o modelo**

Com a biblioteca construída e constantemente atualizada, aliado as ferramentas de transformação geométrica de deslocamento inseridas no Blender, pode-se conseguir com melhor conforto a construção e avaliação de propostas de *layout* industrial.

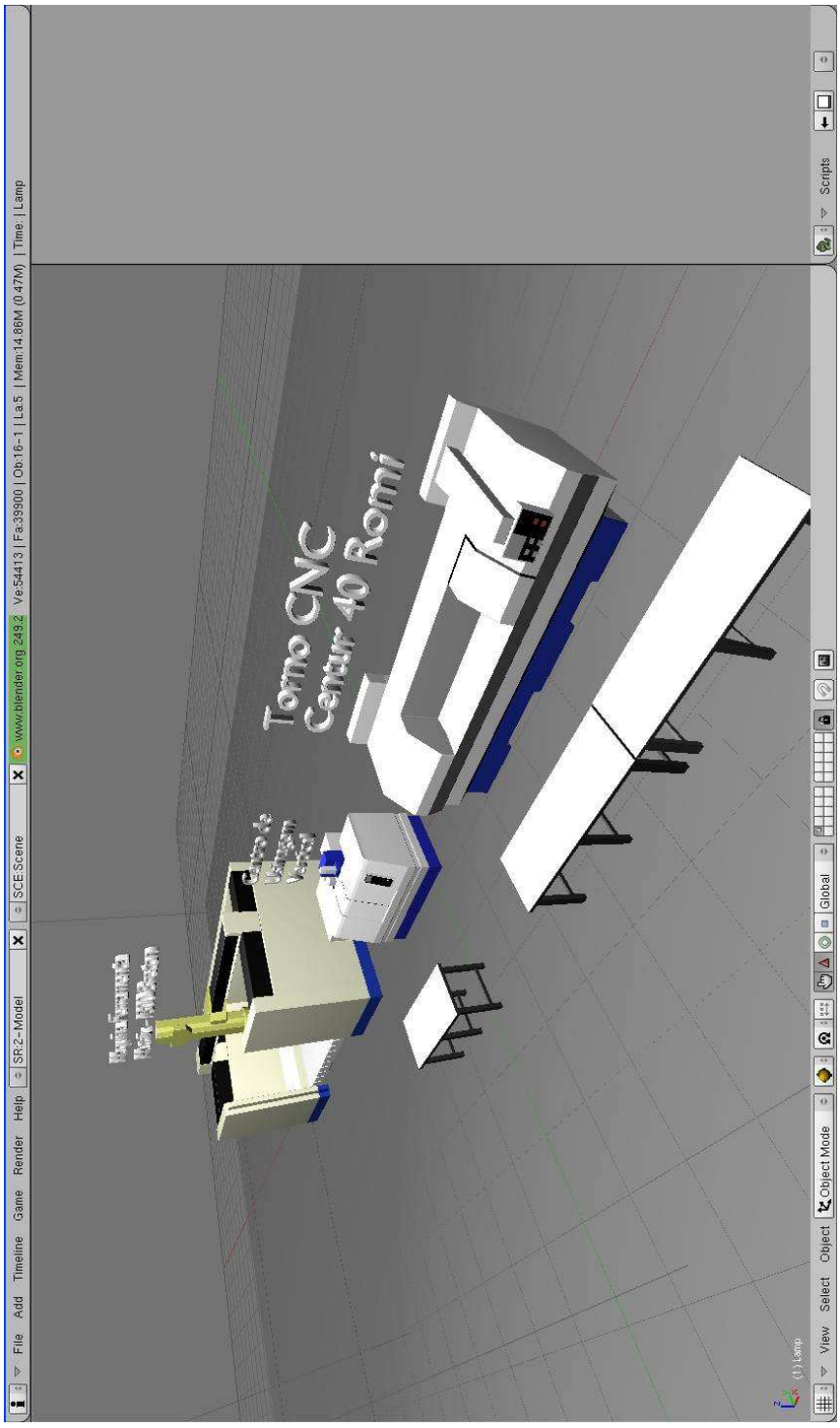


**Figura 16 – *Layout* de célula de manufatura.**

# 6 CONCLUSÃO

As aplicações de modelagem tridimensional podem trazer diversos benefícios em termos de agilidade em projetos de desenvolvimento em diversas áreas. A Arquitetura, Engenharia Mecânica, Civil e Aeronáutica e outras áreas como a medicina há tempos vêm colhendo frutos da melhoria na experiência de se estudar, projetar e desenhar proporcionado pelo mundo virtual da computação gráfica. Mundo este onde errar é permitido e o custo do erro é pequeno.

Partindo disso, seria natural propor para a engenharia de produção, ferramentas que possam melhorar a experiência de se projetar os arranjos físicos, postos de trabalho, instalações de indústria. O campo a ser explorado é imenso, pois muitas técnicas e conceitos parecem se mostrar possível de serem migrados para a computação. Já existem diversos trabalhos relatando e propondo método de construção de *software* voltado a essa questão.

Este trabalho tratou de um item do exposto no parágrafo acima, os arranjos físicos das instalações industriais, partindo do princípio do reuso de *software* e clamando que *software* já existe e algumas características deles os tornam reutilizável para aplicação diversa ao que foi originalmente concebido. Algumas ferramentas foram criadas e incorporadas ao *software* de modelagem e animação tridimensional com o intuito de testar a possibilidade das técnicas e conceitos vistos na literatura funcionarem no ambiente computacional e aferir o esforço do trabalho de implementação.

Quanto a possibilidade, as técnicas elencadas ao longo do trabalho, como ROC, algoritmos genéticos e busca tabu mostraram poder serem inseridas no *software*. Porém tem-se a mais importante ressalva. O esforço para isso é muito grande e uma equipe de desenvolvedores aliados a engenheiros e projetistas é fundamental para se ter o resultado positivo em curto tempo, no que se refere aos algoritmos citados acima, o recurso humano neste trabalho foi um definitivo fator limitante.

Quanto às ferramentas auxiliares, como a de medição (*Caliper*), alinhamento, requer menor esforço e foram implementadas no *software* com a intenção de se mostrar o procedimento de implementação de funções e componentes ao mesmo.

Outra importante característica do trabalho de desenvolvimento de soluções de *layout* industrial é a necessidade de hardware com velocidade de processamento e capacidade de

memória principal crescente à medida que vai aumentando o tamanho da planta industrial e também o desejo por maiores detalhes ou refinamento artístico dos modelos tridimensionais. Portanto o poder de processamento e armazenagem em memória primária é um fator limitante no desenvolvimento de *layout* industrial. Porém, deve-se salientar que esse fator é limitante para qualquer *software* de modelagem tridimensional, das mais diversas áreas. Há estudos em andamento propondo para a computação tridimensional o uso intenso de processador e memória gráfica presente em placas gráficas atualmente comercializadas devido a força do processamento paralelo.

Sobre a opção pelo *software*, temos que considerar alguns pontos críticos e limitações. Inicialmente o *software* que foi a opção neste trabalho foi concebido para usuários técnicos com alto conhecimento e envolvimento com os fluxos de trabalho na empresa que utilizada o Blender, a saber, Not a Number (NaN) e Studio Neo-Geo. Devido a isso, uma grande dificuldade do programa é sua interface gráfica muito baseada em texto. Ao longo tempo em que passou para a iniciativa livre, foi discutido pela comunidade de desenvolvedores e artistas a melhoria no quesito interface gráfica. Atualmente, a versão beta do *software* (2.54), tem a missão de superar esse obstáculo tornando a usabilidade do *software* mais agradável para artistas.

## 6.1 Propostas para trabalhos futuros

Como propostas futuras, além de seguir no desenvolvimento de ferramentas para automatizar o posicionamento de modelos tridimensionais no ambiente virtual 3D e na construção de máquinas e estações de trabalho para povoar a biblioteca de equipamentos, um próximo problema a ser focado seria análise da ergonomia e interação humana com estações de trabalho com o objetivo de melhorar o design e propostas de projetos de estações de trabalho.

BIBLIOGRAFIA

ADEL EL-BAZ, M. (2004). *A genetic algorithm for facility layout problems of different manufacturing environments*. ***Computers & Industrial Engineering***, v.47, n.2-3, p.233-246.

ARGOUD, Ana R.T.T. (2007). **Procedimentos para projeto de arranjo físico modular em manufatura através de algoritmos genéticos de agrupamento**. Tese (Doutorado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2007.

BRITO, A. **Blender 3D: guia do usuário**. 2 ed. São Paulo: Editora Novatec; 2007.

DAVIS, Mark M.; AQUILANO, Nicholas J.; CHASE, Richard B. **Fundamentos da administração da produção**. 3 ed. Porto Alegre: Editora Bookman; 2001.

GAITHER, Norman; FRAZIER, Greg. **Administração da produção e operações**. 8 ed. São Paulo: Editora Thomson; 2002.

LORINI, José F. **Tecnologia de grupo e organização da manufatura**. Florianópolis: Editora da UFSC; 1993.

MARTINS, Petronio G.; LAUGENI, Fernando Piero. **Administração da Produção**. 2 ed. São Paulo: Saraiva, 2005.

MARTINS et al. – Otimização de *Layout*s Industriais com Base em Busca Tabu. **GESTÃO & PRODUÇÃO**, v.10, n.1, p.69-88, abr. 2003.

MULLEN, T. – ***Mastering Blender***. Indianapolis: Wiley Publishing, Inc.; 2009

PRESSMAN, Roger S.. **Engenharia de *Software***. 6. ed. São Paulo: Mcgraw-hill, 2006.

SOMMERVILLE, Ian. **Engenharia de *Software*.** 8. ed. São Paulo: Addison Wesley, 2007.

TIBERTI, A. J. (2003). **Desenvolvimento de *Software* de Apoio ao Projeto de Arranjo Físico de Fábrica Baseado em um Framework Orientado a Objetos**. Tese (Doutorado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2003.

TORRES, Isaías. **Integração de Ferramentas Computacionais Aplicadas ao Projeto e Desenvolvimento de Arranjos Físico de Instalações Industriais**. Dissertação (Mestrado em Engenharia de Produção), 2001. Programa de Pós-graduação em Engenharia de Produção. UFSCar. São Carlos.