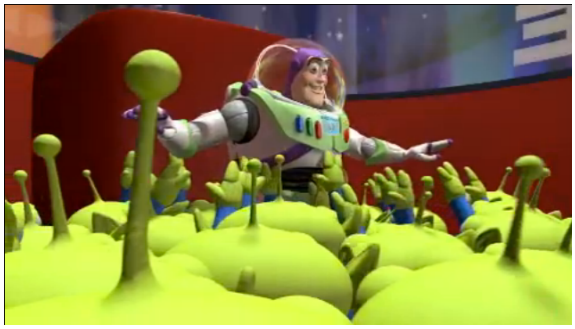
Universidade Federal do Rio Grande do Norte
Departamento de Informática e Matemática Aplicada

Computação Gráfica I – DIM0451

◁ Terceiro Projeto de Programação ▷

The Claw



"— Greetings. I'm Buzz Lightyear. I come in peace."

Apresentação

O objetivo desta projeto de programação é desenvolver uma aplicação gráfica interativa tridimensional (3D). O desenvolvimento desta aplicação vai proporcionar a oportunidade de estudar e utilizar conhecimentos sobre: a) modelagem simples de objetos 3D; b) manipulação hierárquica de cena; c) aplicação de texturas; d) controle interativo e automático de câmera; e) utilização de modelo de iluminação local; f) técnicas para otimização de cena a nível de aplicação (antes do *pipeline*), e; g) utilização de características avançadas do OpenGL.

1 Introdução

Sua tarefa consiste em projetar e desenvolver uma aplicação 3D interativa utilizando OpenGL. A aplicação consiste em um simulador de uma garra robótica—**The Claw**—projetada para mover objetos de uma localização para outra. O ambiente desta aplicação pode ser organizado na forma de um jogo interativo, no qual o jogador tem um determinado limite de tempo para mover as peças do jogo. A cada nível, pode-se dificultar o jogo de várias maneiras: adicionando-se mais objetos para serem movidos; reduzindo o tempo total de movimentação; colocando barreiras entre as localizações 'origem' e 'destino', de maneira a exigir do jogador uma movimentação mais complexa da garra; ou simplesmente desligando as luzes do ambiente e habilitando uma única luz do tipo *spot* (lanterna) atrelada a ponta da garra.

Sugere-se que o trabalho seja executado por uma equipe de até 3 pessoas, e que as tarefas sejam distribuídas de maneira equilibrada, dentro do possível. As tarefas estão organizadas na forma de funcionalidades *básicas* e *complementares*, as quais devem devem ser adicionadas à aplicação.

O trabalho pode ser desenvolvido em qualquer plataforma (preferencialmente linux), desde que seja feito em OpenGL. Espera-se que no final seja produzido um *website* para descrever o trabalho, explicando seus objetivos e, principalmente, destacando como cada uma das funcionalidades básicas e complementares foram incorporadas ao trabalho. O *website* também deve disponibilizar o código fonte (comentado) com instruções claras sobre como baixar, compilar, instalar e utilizar o sistema desenvolvido.

2 Anatomia de uma Garra

"— Who is in charge here? — The claaaaaaaw!"

De acordo com Groove *et al.* [1] um membro robótico é composto de *base*, *braço* e *punho*. O conjunto punho-braço — denominado de *corpo* — é ligado a base, que por sua vez é fixada ao chão, parede ou teto. O braço é fixado à base por um lado e ao punho pelo outro. O braço consiste de elementos denominados *elos* unidos por *juntas* de movimento relativo, onde são acoplados os acionadores para realizarem estes movimentos individualmente de acordo com instruções do sistema de controle. O braço efetua movimentos no *volume de trabalho* do sistema de maneira a posicionar o punho. O punho consiste de várias juntas próximas entre si, que permitem a orientação do *órgão terminal* nas posições que correspondem à tarefa a ser realizada. O órgão terminal é escolhido de acordo com a aplicação, podendo, por exemplo, ser uma **garra**. Os elos do braço são de grande tamanho, para permitir um longo alcance. Por outro lado, os elos do punho são pequenos, e, às vezes, de comprimento nulo, para que o órgão terminal desloque-se o mínimo possível durante a orientação do punho. Numa junta qualquer, o elo que estiver mais próximo da base é denominado de *elo de entrada*; já o elo que estiver mais próximo do órgão terminal é denominado de *elo de saída*.

Em resumo, a base sustenta o corpo, que movimenta o braço, que posiciona o punho, que orienta o órgão terminal (garra), que executa a ação. Em geral utilizam-se 3 juntas para o braço e de 2 a 3 juntas para o punho. A Figura 1 apresenta a anatomia de um membro robótico.

2.1 Par Cinemático e Juntas

Define-se um *par cinemático* como uma conexão entre dois corpos que impõem restrições sobre seus movimentos relativos. Esta conexão é definida por uma junta, que pode ser de vários tipos:

- **Junta rotacional**: também conhecida como *dobradiça*, requer que *(i)* uma linha no corpo em movimento permaneça colinear com uma linha no corpo fixo, e *(ii)* um plano

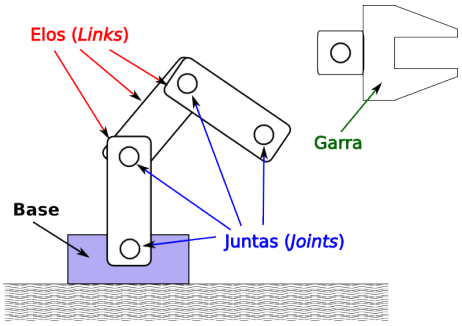


Figura 1: Anatomia de um membro robótico.

perpendicular a esta linha no corpo em movimento mantenha contato com um plano perpendicular similar no corpo fixo. Este tipo de junta possui 1 grau de liberdade e pode ser subdividida de acordo com as direções dos elos de entrada e de saída em relação ao eixo de rotação:

  - *Rotativa de torção* ou *torcional* T: os elos de entrada e de saída têm a mesma direção do eixo de rotação da junta.
  - *Rotativa rotacional* R: os elos de entrada e saída são perpendiculares ao eixo de rotação da junta.
  - *Rotativa revolvente* V: o elo de entrada possui a mesma direção do eixo de rotação, mas o elo de saída é perpendicular ao elo de entrada.

- **Junta prismática**: também conhecida como *deslizadora*, requer que *(i)* uma linha no corpo em movimento permaneça colinear com uma linha no corpo fixo, e *(ii)* um plano paralelo a esta linha no corpo em movimento mantenha contato com um plano paralelo similar no corpo fixo.

A Figura 2 ilustra os quatro tipos de juntas—i.e. as três rotacionais e a prismática. Outros tipos de juntas existem, como *cilíndrica*, *esférica* e *planar*. Contudo, apenas as três rotacionais e a prismática serão necessárias para este trabalho.

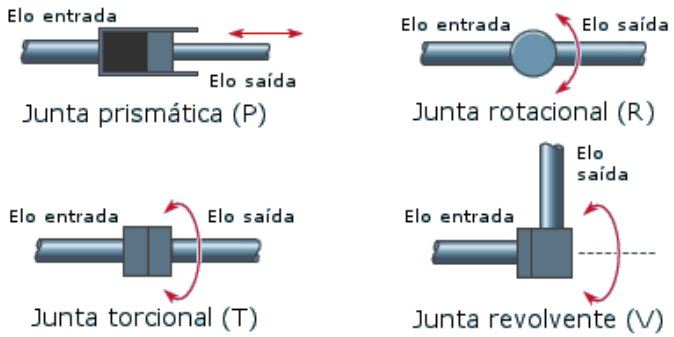


Figura 2: Tipos de juntas rotacionais e prismática. Fonte: Internet.

Os membros robóticos industriais, em geral, podem ser confeccionados a partir deste conjunto de juntas e elos. Para evitar ambiguidade, deve-se buscar uma geometria onde os elos sejam formados por, no máximo, dois segmentos lineares. Por exemplo, a Figura 3 apresenta três braços robóticos determinados, respectivamente, pelas configurações TVR, VRR e TRR

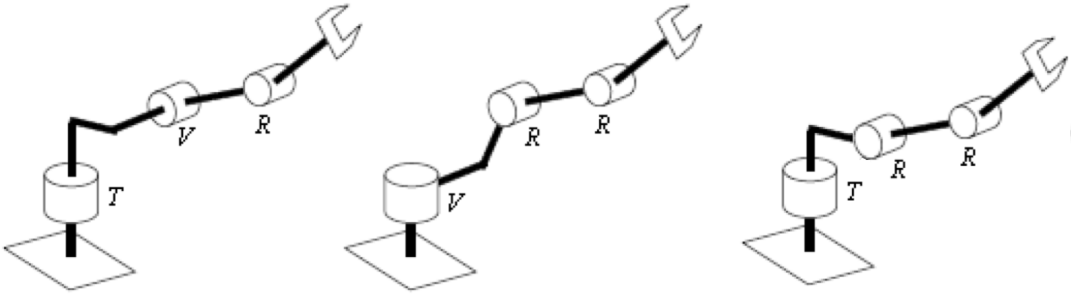


Figura 3: Braços robóticos com configurações TVR (esquerdo), VRR (centro) e TRR (direita). Fonte: Internet.

2.2 Configuração do Robô

A configuração de um robô está relacionada com os tipos de junta que o constituem. Cada configuração pode ser representada por uma notação de letras, sendo especificado as juntas da base até o punho. Assim, um robô TRR teria a junta da base torcional e as duas seguintes (braço) seriam do tipo rotacional. O punho pode ser especificado da mesma forma, separando-se por ":", como por exemplo $TRR{:}RR$. Configurações típicas de robôs industriais estão descritas na Tabela 1.

Tabela 1: Tipos de robôs industriais.

Configuração da base + braço	Símbolos	Configuração do pulso	Símbolos
Cartesiana	LLL	Pulso de 2 eixos	RT
Cilíndrica	LVL	Pulso de 3 eixos	TRT
Articulada ou revoluta	TRR		
Esférica	TRL		
SCARA	VRL		

3 A Garra

Neste projeto de programação a equipe deve implementar um braço robótico do tipo *articulado*, sendo o tipo de pulso livre. Veja na Figura 4 exemplos de robôs nesta configuração.

Figura 4: Exemplos de robôs articulados. Fonte: Internet.

Na ponta, pode-se utilizar uma barra com 2 ou 3 dedos, movidos por junta prismática ou rotacional, conforme exemplos ilustrados na Figura 5.

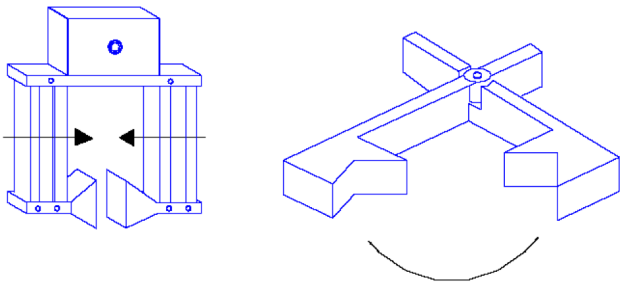


Figura 5: Exemplos de garras. Fonte: Internet.

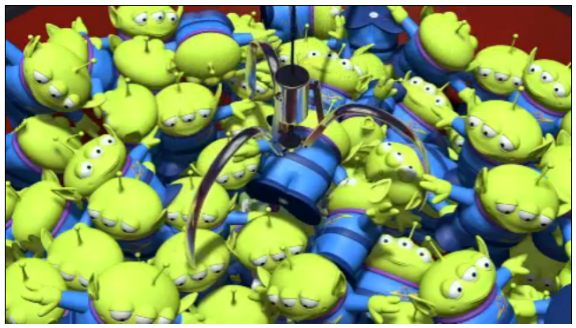
3.1 Agarrando objetos

"— The claw is our master! The claw chooses who will go and who will stay."

A realização da ação de agarrar objetos deve ser baseado na colisão entre as faces da garra (órgão terminal) e do objeto em questão. Para facilitar a implementação da simulação, basta apenas que todos os dedos da garra entrem em contato com o objeto para que seja considerado "agarrado" e, portanto, passível de ser erguido no ambiente virtual.

Cuidado deve ser tomado para evitar que os objetos atravessem uns aos outros, inclusive a *autointerseção* dos robô. Autointerseção pode ser evitada de maneira simples, aplicando-se restrições aos ângulos de rotação de cada junta. Porém, a interseção com o "piso" ou "paredes" do ambiente virtual, bem como com os objetos, deve ser precisamente calculada.

3.2 Movimentação



"— Shuuuuush. The claw. It moves!"

O membro robótico proposto neste projeto é uma *cadeia cinemática*, ou seja, uma montagem de vários *pares cinemáticos* (juntas) conectando segmentos de corpos rígidos (elos).

O posicionamento do órgão terminal em determinado ponto do espaço tridimensional (3D) pode ser realizado por cinemática *direta* ou *inversa*. Na cinemática direta, deve-se determinar a posição do órgão terminal (*end effector*) em função dos ângulos das juntas que compõem o robô. Em outras palavras, resolve-se o seguinte problema: *"Dados as posições das juntas e seus ângulos, qual a posição do órgão terminal?"* A solução para este problema é única.

Mais formalmente: considere uma cadeia serial de n elos, sendo Θ_i o ângulo do elo i. Dados $\Theta_0 \dots \Theta_n$, o frame do elo n relativo ao elo 0 é:

$$^0T_n = \prod_{i=1}^{n} {}^{i-1}T_i(\Theta_i) \tag{1}$$

onde $^{i-1}T_i(\Theta_i)$ é a matriz de transformação que leva do frame do elo i para o frame do elo $i-1$. Particularmente em robótica, utiliza-se os parâmetros *Denavit-Hartenberg* para descrever estas transformações, que nada mais são do que as nossas velhas conhecidas matrizes de transformações geométricas.

Já a cinemática inversa se baseia em um cenário mais complexo no qual tem-se a posição inicial e final do órgão terminal e procura-se deduzir qual a variação dos ângulos das juntas do robô necessárias para mover da configuração inicial para a final. Este procedimento é muito comum em animação de personagens, na qual o animador especifica dois pontos chaves da pose de um personagem (*key frame*) e o sistema de animação realiza a interpolação de um *key frame* para o outro, criando as poses intermediárias para tal movimentação.

3.3 Visualização

Um dos problemas a serem resolvidos no projeto está relacionado com a visualização do ambiente. É fácil perceber que ao apresentar um ambiente 3D através de uma projeção 2D (a tela) surge o problema de fornecer ao usuários "pistas" visuais acerca da localização relativa da garra e objetos no ambiente. Em outras palavras, como comunicar ao usuário informações de profundidade, ou seja, sinalizar que a garra está posicionada acima, a frente, ao lado, ou atrás de um determinado objeto?

A solução para este problema necessariamente envolverá a manipulação de câmera para permitir visões da cena de pontos de vista distintos. A transição de um ponto de vista para outro deve ser suave e contínua, de maneira que o usuário não tenha sua imersão comprometida e não se perca dentro do ambiente.

Em complemento ao método de manipulação de vários pontos de vista, o sistema deve também oferecer algum tipo de feedback visual no ambiente. Isso pode ser disponibilizado na forma de sombras da garra sobre o piso/objetos ou simplesmente um cursor 3D na forma de 3 linhas mutualmente perpendiculares irradiando da garra (origem) até interseccionar com as as paredes, piso e teto (i.e. 3 eixos ortogonais).

3.4 Interface

Um dos grandes desafios deste trabalho é, justamente, conseguir acomodar todos os parâmetros de manipulação do robô mais os parâmetros de manipulação da câmera em uma interface intuitiva e funcional.

Lembre-se que partimos do princípio que o ambiente de simulação será exibido na forma de um jogo eletrônico, no qual o jogador (usuário) deve mover objetos de um local para outro no menor tempo possível, desviando a garra de obstáculos que possam existir entre as localizações origem e destino. Neste sentido, é fortemente recomendável a incorporação de um dispositivo de interação não convencional, como por exemplo um *joystick* ou *joypad*.

Além da manipulação do membro robótico, da mudança de ponto de vista da câmera, a aplicação deverá permitir a manipulação interativa de luzes no ambiente. Isto pode ser oferecido ao usuário na forma de interruptores que ligam/desligam luzes com configurações diferentes. Pede-se que sejam utilizadas luzes/material difuso e especular, no mínimo.

3.5 Gameplay

Espera-se um jogo com pelo menos uma fase, na qual o jogador deve mover pelo menos 3 objetos de uma área para outra, no menor tempo possível.

Desta forma o jogo deve ser capaz de medir e exibir o tempo de execução. Deve-se oferecer 3 objetos-alvo distintos, como por exemplo, esferas e cubos. Os obstáculos podem ser paredes, cilindros, pitâmides, etc., de maneira que impeçam uma movimentação direta da garra de um local para outro. Uma maneira simples de se criar um obstáculo é colocar, por exemplo, a área detino em uma plataforma mais elevada (ou rebaixada) em relação a área de origem.

Sempre que a garra colidir com elementos do cenário (obstáculos, outros objetos, paredes, piso, etc.) deve-se penalizar o jogador tirando-se pontos OU fazendo com que a garra solte o objeto. No último caso o jogador teria que agarrar o mesmo objeto novamente. Para facilitar a implementação, não é necessário realizar animação da queda do objeto de maneira realista, ou seja, baseado em Física—uma animação com velocidade constante é o suficiente. Claro que os projetos que realizarem as interações baseados em Física receberão pontos extras.

Um a outra maneira de aumentar o grau de dificuldade em uma determinada fase é utilizar luzes da seguinte forma: o cenário é iniciado na escuridão total, habilitando-se apenas uma luz *spot* na garra, apontando para baixo. Desta maneira, o jogador terá que movimentar a garra pelo ambiente para poder localizar tanto os objetos quanto a área de destino dos objetos.

O ponto mais importante aqui é ser criativo na elaboração de obstáculos, definição da interface, manipulação de câmera, enfim, na apresentação geral do jogo. É esta criatividade que vai diferenciar um projeto de outro e tornar o jogo mais (ou menos) interessante.

4 Funcionalidades do Sistema

O Sistema a ser desenvolvido deverá oferecer as seguintes funcionalidades:

B1: **Modelagem do robô.** Sua aplicação deve modelar um membro robótico com, pelo menos, 1 base, 3 juntas para o conjunto base+braço, 2 juntas para o pulso e 1 garra. Os componentes do robô devem ser compostos a partir de unidades simples de elos e juntas, cuja topologia e funcionamento foram previamente explicados na Seção 2. Pontos serão atribuídos de acordo com a qualidade da modelagem do braço. Braços simples receberão menos pontos do que braços visualmente mais sofisticados.

B2: **Animação do robô.** A aplicação deve fazer uso de cinemática direta para posicionar a garra sobre os objetos, sendo capaz de atingir qualquer ponto dentro do volume de trabalho. A animação deve ser baseada em transformações geométricas hierárquicas.

B3: **Animação de câmera.** Como indicado na seção anterior, o sistema deve oferecer a possibilidade de realizar uma movimentação de câmera para permitir a identificação da localização relativa da garra em relação aos objetos. Esta movimentação pode ser interativa (preferível) ou realizada de maneira automática para pontos chaves pré-estabelecidos.

Neste último caso, a animação **deve** ser suave. Animações que simplesmente alternam de um ponto de vista para outro não receberão a pontuação completa referente a este item.

B4: **Interação com usuário.** A aplicação deve suportar interação do usuário com o sistema através de teclado e/ou mouse (mínimo). Alternativamente, podem ser utilizados dispositivos não convencionais, como celular, joystick, webcam, joypad, etc.

B5: **Iluminação e tonalização suave.** O sistema deve oferecer, pelo menos, duas luzes pontuais. O sistema de iluminação deve possibilitar a combinação de luz/material para demonstrar o efeito de iluminação apenas ambiente, ou difusa, ou especular e a combinação dos três componentes. Adicionalmente, deve-se criar uma terceira luz *spot* atrelada à garra.

B6: **Mapeamento de textura.** O sistema deve implementar mapeamento de textura para pelo menos um dos objetos tridimensionais da cena.

B7: **Detecção de Colisão.** O sistema deve suportar, pelo menos, a colisão por *bounding volume*. Contudo, para o sistema se tornar visualmente convincente, é importante dedicar tempo para implementar a detecção completa, i.e., planos contra planos (tópico **C3**, explicado mais adiante).

Além das funcionalidades acima mencionadas, sua aplicação deve adicionar algumas das técnicas avançadas de computação gráfica listadas a seguir. Obviamente, para aumentar o nível de aprendizado aconselha-se que sejam adicionadas o maior número de funcionalidades possível. Todo o esforço extra poderá ser recompensado na forma de pontos extras.

Note que as funcionalidades complementares somente serão consideradas se todas as funcionalidade básica tiverem sido implementadas. As funcionalidades complementares devem ser listadas (e demonstradas) claramente no *website* do projeto, de maneira a facilitar a avaliação do trabalho.

C1: **View frustum culling.** Em ambientes 3D complexos, é necessário limitar o número de primitivas 3D que são enviadas para o *pipeline* de renderização com intuito de manter uma taxa de renderização que caracterize uma aplicação interativa em tempo real (i.e. acima de 15 *frames* por segundo ou 15Hz). Uma forma simples de reduzir a quantidade de primitivas enviadas ao *pipeline* consiste em não desenhar os objetos que estejam fora do volume de visão (*frustum*) da câmera. Uma alternativa mais sofisticada consiste em computar *a priori* volumes envolventes para os objetos da cena e então testar se estes volumes (por exemplo uma esfera) intersecta o volume de visão antes de desenhar seu conteúdo (i.e. o objeto envolvido). Por fim, é possível aplicar técnicas mais complexas e de maior eficiência que organizam o espaço e seus objetos (e seus volumes envolventes) em uma estrutura no estilo de árvore, denominado de *grafo de cena hierárquico*. Exem-

plos desta última opção são técnicas como *octrees* ou *binary space partitioning trees* (BSP *trees*).

C2: **Controle autônomo de câmera.** É possível controlar a câmera de tal maneira que a mesma sempre enquadre a garra e o objeto alvo, similar ao que ocorre em jogos exibidos em terceira pessoa. Para tanto, o sistema deve ser capaz de alterar tanto a localização da câmera no espaço como também sua orientação, tudo de maneira automática e em função das posições da garra e do objeto-alvo dentro ao ambiente 3D.

C3: **Detecção de colisão refinada.** Melhorar o processo de detecção de colisão, realizando cálculos de interseção entre cada uma das faces do objeto e da garra.

C4: **Simulação dinâmica.** Pode ser interessante incluir comportamentos dinâmicos e baseado em Física para objetos da cena virtual. Por exemplo, ao soltar um cubo poderíamos observar o cubo rolando, como resultado do cubo ter colidido com o chão de quina. Este cubo, por sua vez, ao cair e rolar pode se chocar com outro objeto da cena, deslocando-o de lugar (reação à colisão). Para que isso seja possível, recomenda-se a utilização de motores de física, como por exemplo a BULLET, ODE, NEWTON, HAVOK, etc.

C5: **On-screen control panel** ou **HUD (Heads Up Display).** Muitas aplicações 3D, como por exemplo vídeo games, reservam uma porção da área de display para oferecer um painel de controle ou exibir informações sobre a aplicação. Por exemplo, em um jogo de corrida de automóveis, o painel do carro localiza-se na parte inferior da área de display e é utilizado para apresentar informações sobre o veículo, como velocidade atual, nível de combustível, etc. Outro uso comum de HUD ocorre em aplicações que exibem um mapa 2D de um cenário 3D no qual o usuário encontra-se imerso e em navegação. Existem várias técnicas para gerar HUDs, como por exemplo o uso de projeção ortográfica conjugado com o *stencil buffer*.

C6: **Efeitos de renderização avançados.** Existem uma série de efeitos visuais suportados pelo OpenGL que podem ser incorporados ao sistema. Por exemplo, transparência, *motion blur*, *lens flare*, *night vision*, *soft shadows*, reflexões, *bump mapping*, profundidade de campo, *environment mapping*, *billboarding*, *skyboxes*, etc. Estes efeitos podem ser obtidos via manipulação dos vários buffers do OpenGL conjugados com várias passagens de renderização para o mesmo *frame*, ou então através da manipulação de imagens e texturas.

C7: **Utilização de shaders.** Desenvolver o sistema por meio de programação de shaders, ao invés de se basear na funcionalidade fixa do pipeline.

5 Avaliação e Entrega

A realização dos itens básicos faz com que o trabalho atinja, no máximo, nota $7,0$ (sete vírgula zero). Para atingir nota dez, é necessário implementar as atividades complementares, cuja pontuação está descrita na Tabela 2.

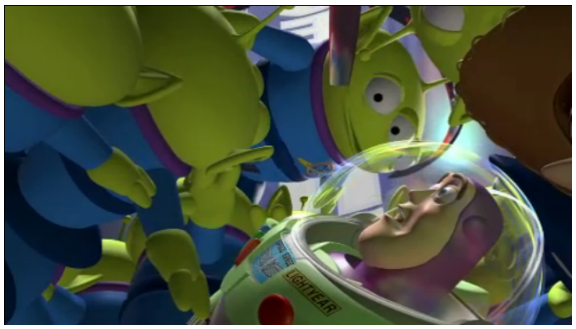
Tabela 2: Tabela com distribuição das pontuações.

Funcionalidades	**Item**	**Parcela da pontuação**
Básica	B1	15%
	B2	10%
	B3	10%
	B4	10%
	B5	10%
	B6	10%
	B7	5%
Complementar	C1 a C4	20% (cada item)
	C5 a C7	10% (cada item)
Total		**100%**
Extra	implementar CX extras	até $+20\%$

Vale ressaltar que a pontuação indicada na Tabela 2 não é definitiva e imutável. Ela serve apenas como um guia de como o trabalho será avaliado em linhas gerais. Desta forma, durante a correção é possível realizar ajustes nas pontuações indicadas visando adequar a pontuação ao nível de dificuldade dos itens solicitados. Por exemplo, se a aplicação utilizar um dispositivos de interação com o usuário não convencional (como webcam com reconhecimento de imagem), a pontuação correspondente deverá ter seu valor aumentado.

Além da entrega do trabalho, a equipe deverá produzir um *website* (não é necessário publicá-lo) para descrever o projeto e disponibilizar o código fonte. No *website* deve ser explicado o *gameplay*, preferencialmente com *screen shots*. Além do *website*, o trabalho deve ser acompanhado de um arquivo `leia.me` no qual são indicados os membros da equipe e os itens básicos e complemetares implementados, bem como eventuais itens extras.

6 Autoria e Política de Colaboração



"— I have been chosen."

O trabalho pode ser realizado individualmente ou em grupos de até três componentes, sendo que qualquer equipe poderá ser convocadas para entrevista. O objetivo da entrevista é duplo: confirmar a autoria do trabalho e determinar a contribuição real de cada componente em relação ao trabalho. Note que os membros da equipe devem ser capazes de explicar, com desenvoltura, qualquer trecho do trabalho solicitado durante a entrevista. Portanto, é possível que, após a entrevista, ocorra redução da nota geral do trabalho ou ajustes nas notas individuais, de maneira a refletir a verdadeira contribuição de cada membro, conforme determinado na entrevista.

O trabalho em cooperação entre alunos da turma é estimulado. Porém, esta interação **não** deve ser entendida como permissão para utilização de código ou parte de código de outras equipes, o que pode caracterizar a situação de plágio. Trabalhos plagiados receberão nota **zero** automaticamente, independente de quem seja o verdadeiro autor dos trabalhos infratores.

Será permitido, contudo, que algumas funções auxiliares, como por exemplo para carregar imagem na memória, ou mesmo carregar modelos tridimensionais em uma estrutura interna, sejam de autoria de terceiros. Contudo, o código auxiliar não deve ultrapassar **10%** (dez por cento) do montante total de linhas de código. Além disso, as funções ou códigos que não forem de autoria da equipe **devem estar claramente identificados tanto no código fonte quanto no website do projeto**, estando sujeito a penalizações (a definir), caso esta condição não seja satisfeita. O prazo de submissão do trabalho será divulgado na Turma Virtual do Sigaa. Submissões por email serão ignoradas e a equipe infratora poderá receber nota zero.

7 Considerações Finais

"— Farewell my friends. I'll go on to a better place."

Este projeto possui vários itens complementares. Este tipo de formato permite que os alunos que possuem um maior interesse pela área de jogos eletrônicos ou computação gráfica possam desenvolver aplicações mais complexas, sendo recompensados por tal iniciativa.

Espera-se que os membros das equipes realizem uma certa parcela de autoaprendizado, especialmente no que diz respeito às funcionalidades complementares. Portanto,

não assuma a postura de ficar esperando que todo o assunto descrito aqui seja apresentado na disciplina para só então iniciar o desenvolvimento do trabalho. Os conceitos gerais sobre sistemas 3D e uso do OpenGL já foram apresentados e são suficientes para o desenvolvimento do trabalho.

Referências

[1] Mikell P. Groove, Mitchell Weiss, Roger N. Nagel, and Nicholas G. Odrey. *Industrial Robotics: Technology, Programming, and Applications*. Mcgraw-Hill College, New York, USA, March 1986.