

UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

Criação de Cenas Interativas Usando Realidade
Aumentada

Daniel Hoffmann Bernardes



São Carlos - SP

Criação de Cenas Interativas Usando Realidade Aumentada

Daniel Hoffmann Bernardes

Orientador: *Moacir Pereira Ponti Junior*

Monografia final de conclusão de curso apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP – para obtenção do título de Bacharel em Ciências de Computação. Área de Concentração: Computação Gráfica.

USP - São Carlos
Maio de 2012

Agradecimentos

À minha família por todo o apoio durante todos esses anos.

Aos meus amigos que dividiram tantos anos a mesma casa comigo.

Aos meus colegas e amigos de classe.

Ao meu orientador por acreditar em mim e na minha ideia.

Resumo

O objetivo desta monografia é explicar o funcionamento de um sistema que usa realidade aumentada. O propósito do sistema é a criação de cenas através do posicionamento de diferentes objetos 3D usando marcadores detectados por uma webcam e técnicas de visão computacional. As cenas criadas podem ser então manipuladas pela aplicação das leis Newtonianas de física nos objetos e observando a interação entre eles. Para criar o sistema foram usadas as bibliotecas ARToolKit para detecção dos marcadores, Bullet para simulação da física e Panda3D para posicionamento de modelos e renderização. Objetos 3D foram criados para serem usados nas cenas, a maioria foi gerada proceduralmente (programando a posição de cada ponto de cada polígono e realizando o mapeamento de texturas manualmente). Neste projeto é mostrado que a utilização de realidade aumentada pode melhorar显著mente a interação humano-computador.

Abstract

The purpose of this dissertation is to describe the operation of a system that uses augmented reality. The purpose of the system is the creation of scenes by positioning different 3D objects using markers detected by a webcam and computer vision techniques. The created scenes can then be manipulated by applying Newton physics laws on the objects and observing the interaction between them. To create the system the following libraries were used: ARToolKit for marker detection, Bullet for physics simulation and Panda3D for positioning and rendering. 3D objects were created to be used in the scenes, most of them were procedurally generated (programming the position of each point of each polygon and doing the texture mapping manually.) In this project it is shown that augmented reality can significantly improve human-computer interaction

Sumário

Lista de Figuras	p. III
1 Introdução	p. 5
1.1 Contextualização e Motivação	p. 5
1.2 Objetivos	p. 5
1.3 Marcadores	p. 6
2 Revisão Bibliográfica	p. 7
2.1 Considerações Iniciais	p. 7
2.2 Realidade Aumentada	p. 7
2.3 Marker tracking and hmd calibration for a video-based augmented reality conferencing system[2]	p. 8
2.4 Developing an Augmented Reality Racing Game [3]	p. 9
2.5 Tangible authoring of 3D virtual scenes in dynamic augmented reality environment[4]	p. 11
2.6 Augmented Reality for Board Games[5]	p. 11
2.7 Motion in Augmented Reality Games: An Engine for Creating Plausible Physical Interactions in Augmented Reality Games[6]	p. 11
2.8 Considerações Finais	p. 12
3 Desenvolvimento do Trabalho	p. 13
3.1 Considerações Iniciais	p. 13
3.2 Projeto	p. 13
3.3 Instalação	p. 14

3.4	Bibliotecas	p. 14
3.4.1	ARToolKit	p. 14
3.4.2	Panda3D[7]	p. 15
3.4.3	Bullet[8]	p. 17
3.5	Cena	p. 19
3.6	Criação dos Objetos	p. 20
3.6.1	Criação de Retângulos Procedimentalmente	p. 21
3.6.2	Mapeamento de Texturas	p. 23
3.6.3	Criação de Corpo Rígido	p. 23
3.6.4	Plano	p. 24
3.6.5	Cubo	p. 25
3.6.6	Esfera	p. 26
3.6.7	Estrada	p. 26
3.6.8	Curva de Estrada	p. 28
3.7	Resultados Obtidos	p. 29
3.7.1	Funcionamento	p. 30
3.8	Dificuldades	p. 31
3.9	Limitações	p. 33
3.10	Considerações Finais	p. 33
4	Conclusão	p. 35
4.1	Contribuições	p. 35
4.2	Trabalhos Futuros	p. 35
4.3	Considerações sobre o Curso de Graduação	p. 36
Referências Bibliográficas		p. 38
Apêndice A - Código Fonte		p. 39

Lista de Figuras

1.1	Marcador Kanji. É usado como chão da cena.	p. 6
1.2	Marcador Traço. É usado para posicionar objetos na cena.	p. 6
2.1	Sistema de Realidade aumentada usando visão indireta	p. 8
2.2	Video Conferência usando Realidade aumentada	p. 9
2.3	AR Racing	p. 10
3.1	Funcionamento da biblioteca ARToolKit	p. 15
3.2	Sistema de coordenadas do Panda3D	p. 16
3.3	Funcionamento da biblioteca Panda3D	p. 18
3.4	Funcionamento da biblioteca Bullet	p. 19
3.5	Árvore de Cena 1	p. 21
3.6	Árvore de Cena 2	p. 22
3.7	Textura de tijolo	p. 23
3.8	Um plano numa cena.	p. 24
3.9	Um cubo numa cena.	p. 25
3.10	Uma esfera numa cena.	p. 26
3.11	Uma estrada numa cena.	p. 26
3.12	Comparação entre estradas de diferentes tamanhos	p. 27
3.13	Uma estrada criada com $L=tamX$	p. 27
3.14	A árvore de uma estrada com <i>guard-rail</i>	p. 28
3.15	Curva de Estrada <i>guard-rail</i>	p. 28
3.16	Árvore de uma Curva de Estrada com <i>guard-rail</i>	p. 29
3.17	Uma cena com vários objetos.	p. 29

3.18 Controladores	p. 30
3.19 Cena criada usando o sistema	p. 31
3.20 A mesma cena, alguns segundos depois	p. 31
3.21 Bug presente no Panda3D no Windows	p. 32
3.22 Percepção de profundidade	p. 33

1 *Introdução*

1.1 Contextualização e Motivação

O objetivo desta monografia é descrever o funcionamento de um sistema de realidade aumentada. A principal área de pesquisa é a Computação Gráfica, mas o sistema também engloba Interação Humano-Computador, Processamento de Imagens, Geometria Computacional e Visão Computacional.

A motivação principal é proporcionar uma forma diferente de interagir com uma cena 3d ao invés de usar somente os tradicionais mouse, teclado e *joystick*. É esperado que este sistema proporcione uma adição significativa à Interação Humano-Computador, quando comparado com os dispositivos de entradas convencionais. Além de um conhecimento mais profundo sobre meios de usuários interagirem com ambientes de três dimensões, a união de bibliotecas de programação e métodos de áreas diferentes como Computação Gráfica e Visão Computacional.

1.2 Objetivos

O principal objetivo do trabalho foi o uso de métodos para proporcionar criação do aumentamento de uma cena real obtida por uma câmera usando modelos 3d posicionados usando marcadores. Após a criação da cena é possível interagir com ela através das leis de movimento da física (por exemplo, se um objeto não estiver encostado no chão ele irá acelerar para baixo com a gravidade). Também é possível interagir usando outros objetos presos à um marcador e não sujeitos às leis da física presentes na cena. Estes objetos não são afetados pela gravidade e não podem ser mudados de posição através de colisões (ficando sempre preso ao marcador).



Figura 1.1: Marcador Kanji.

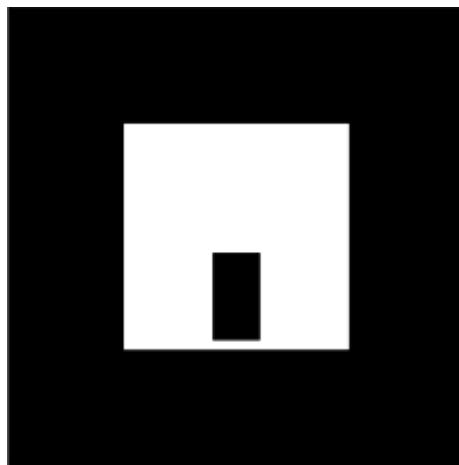


Figura 1.2: Marcador Traço.

1.3 Marcadores

Os marcadores são quadrados preto e branco com padrões no centro que podem ser detectados mais facilmente usando visão computacional. A Figura 1.1 mostra um dos marcadores utilizados. Os marcadores podem ser impressos em qualquer impressora a jato de tinta.

No sistema são usados dois marcadores, um com o padrão kanji no centro (Figura 1.1) é usado como o chão da cena e o outro padrão que aqui será chamado de traço (Figura 1.2) é onde o objeto selecionado pelo usuário estará preso.

2 *Revisão Bibliográfica*

2.1 Considerações Iniciais

Neste capítulo será discutido o conceito de realidade aumentada e como ela pode mudar o meio que as pessoas conseguem interagir com o computador assim como discutir alguns trabalhos da área

2.2 Realidade Aumentada

O conceito de realidade aumentada pode ser explicado com boa clareza ao comparar realidade aumentada e realidade virtual, como Kirner e Tori colocaram [1]: "Diferentemente da realidade virtual, que transporta o usuário para o ambiente virtual, a realidade aumentada mantém o usuário no seu ambiente físico e transporta o ambiente virtual para o espaço do usuário, permitindo a interação com o mundo virtual, de maneira mais natural e sem necessidade de treinamento ou adaptação."

É notável que a realidade aumentada depende de um meio de captar o ambiente físico. No caso do sistema apresentado aqui, este meio é uma câmera. A interatividade do sistema é proporcionada pelo ambiente virtual onde objetos virtuais podem ser manipulados. Sistemas mais complexos podem usar um *head-mounted display*¹, mas não é este o caso para este sistema.

Kirner e Tori também dividem os sistemas de realidade aumentada em dois tipos: visão direta e visão indireta. De acordo com os autores [1]:

"Na visão direta, as imagens do mundo real podem ser vistas a olho nu ou trazidas, através de vídeo, enquanto os objetos virtuais gerados por computador podem ser projetados nos olhos, misturados ao vídeo do mundo real ou projetados no cenário real. Na visão indireta, as imagens do mundo real e do mundo virtual são misturadas em vídeo e mostradas ao usuário."

¹*head-mounted display* é um dispositivo de vídeo usado na cabeça que possui uma ou duas telas em frente aos olhos



Figura 2.1: Sistema de Realidade aumentada usando visão indireta. Fonte: [1]

O sistema apresentado aqui usa visão indireta como apresentado pela Figura 2.1. Este tipo de sistema foi usado por ser mais prático de ser criado e por precisar apenas de uma câmera USB.

2.3 Marker tracking and hmd calibration for a video-based augmented reality conferencing system[2]

Este artigo foi publicado em 1999 e apresenta um sistema de realidade aumentada para vídeo-conferência onde os vídeos das pessoas presentes na conferência são colocadas em cima de marcadores e o usuário consegue mudar a localização de cada pessoa mudando a posição dos marcadores. Este sistema utiliza um *head-mounted display* para o usuário principal. A Figura 2.2 mostra o sistema em funcionamento.

Além de simples vídeo-conferência, o sistema também proporciona um quadro virtual onde qualquer um dos usuários presentes na vídeo-conferência pode escrever usando uma caneta. Esta caneta possui um LED que é ativado pela pressão de tocar a caneta no quadro, quando a câmera presente no *head-mounted display* detecta o LED o sistema estima a posição da caneta em relação ao quadro e cria um traço virtual onde a caneta passa. O conteúdo deste quadro é visível para todos os outros usuários possibilitando assim a troca de anotações e outras informações.



Figura 2.2: Video Conferência usando Realidade aumentada. Fonte: [2]

O artigo entra em grandes detalhes técnicos do uso de realidade aumentada com marcadores. Entre estes detalhes estão: Detecção dos marcadores usando técnicas de visão computacional, calibramento de câmera, precisão da detecção de marcador, detecção da posição da caneta usando o LED presente nela, vantagens do uso de realidade aumentada para conferências e troca de informações

Este trabalho eventualmente gerou a biblioteca ARToolKit usada no sistema apresentado nesta monografia.

2.4 Developing an Augmented Reality Racing Game [3]

Este sistema desenvolvido em 2008 é similar ao desenvolvido aqui, mas difere por usar um sistema multi-marcador para o chão e um *head-mounted display* (visão direta). Tal sistema multi-marcador usa vários marcadores com posições fixas entre si, enquanto um destes marcadores permanecer dentro da visão da câmera o sistema não perde o ponto de referência para o resto da geometria. O sistema também simula a montagem de uma pista e um simulador de corrida com suporte a mais de um jogador. A Figura 2.3 mostra o multi-marcador e o usuário interagindo com o jogo.

O escopo do sistema apresentado aqui é mais limitado por este ser um projeto de graduação e individual. O jogo de corrida apresentado foi feito por uma equipe de quatro pessoas e possi-

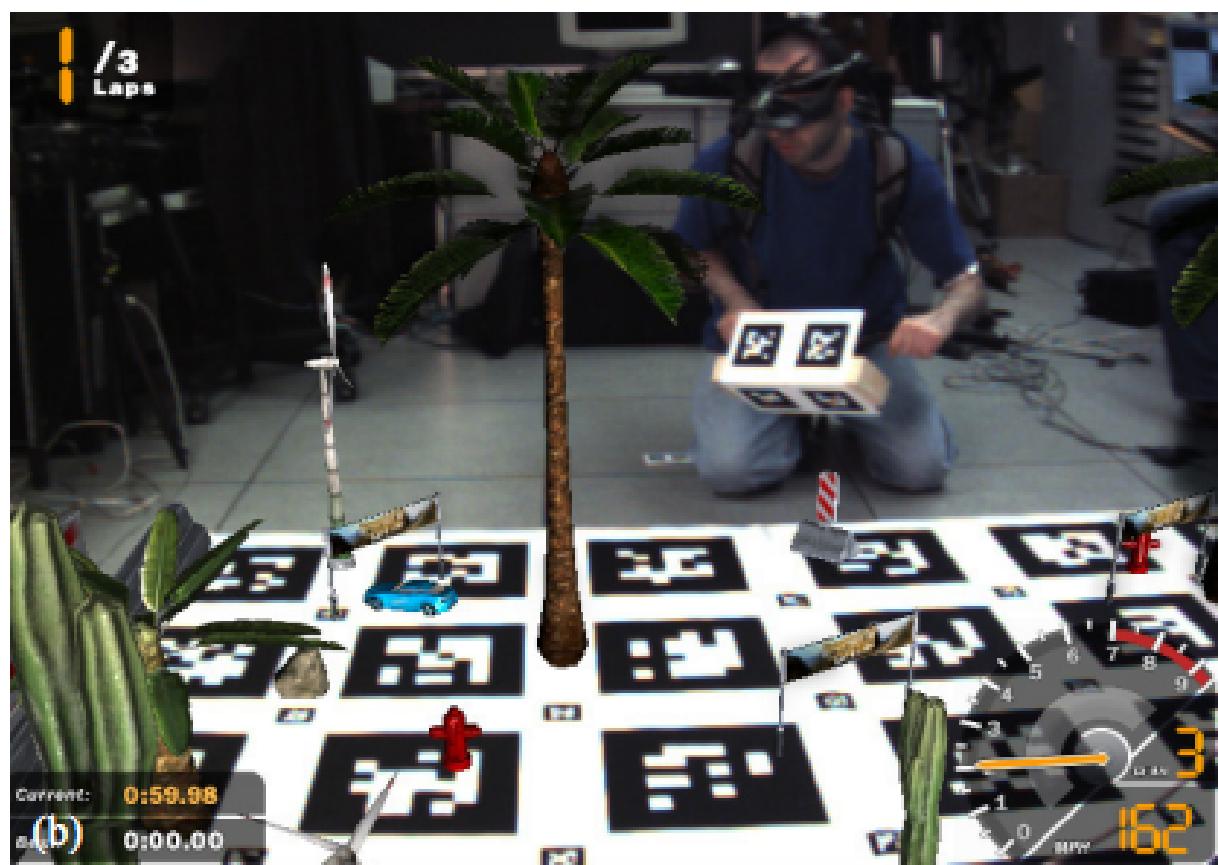


Figura 2.3: Jogo de corrida usando realidade aumentada. Fonte: [3]

velmente com um tempo maior de desenvolvimento do que um semestre.

2.5 Tangible authoring of 3D virtual scenes in dynamic augmented reality environment[4]

Este sistema é outro similar ao desenvolvido aqui, mas difere por prover um sistema de cadastramento de objetos usando RFID², objetos cadastrados podem ser mapeados em modelos 3D (idealmente uma representação 3D do próprio objeto cujo o RFID foi lido). Possibilitando o uso de objetos reais para a escolha do objeto a ser posicionado na cena. Além disso o sistema também suporta o posicionamento de vídeos em cartões que podem ser posicionados na cena usando os marcadores.

Este sistema usa realidade aumentada para criação das cenas, mas depois de criada a interação usando realidade é mínima ao contrário do sistema apresentado aqui.

2.6 Augmented Reality for Board Games[5]

Este sistema exemplifica o uso de realidade aumentada para introduzir elementos virtuais em jogos de tabuleiro. No artigo os autores usaram o jogo *Monopoly* (Banco Imobiliário no Brasil), o objetivo é aumentar a imersão do jogo proporcionando uma experiência mais rica, entretanto ainda é uma experiência diferente de jogos eletrônicos tradicionais.

Neste sistema as peças normalmente presentes no tabuleiro (os peões, os hoteis, entre outros) são substituídas por modelos 3D. O sistema não usa marcadores tradicionais, ao invés ele usa o conhecimento prévio do formato do tabuleiro para posicionar os modelos.

2.7 Motion in Augmented Reality Games: An Engine for Creating Plausible Physical Interactions in Augmented Reality Games[6]

Ao contrário dos sistemas anteriores, este sistema é bastante focado na integração entre a física da cena e a física real para aumentar a imersão do usuário. Este artigo foca mais no uso de realidade aumentada com simulações de física para o uso em jogos eletrônicos. O autor

²RFID é um sistema para identificação usando ondas de rádio. É semelhante à código de barras, mas não requer visão direta do leitor.

explica: "Jogos de realidade aumentada requerem simulações de física sofisticadas, já que o propósito de qualquer aplicação de realidade aumentada é criar a ilusão que objetos reais e virtuais coexistem. Isto requer a adição de movimentos de física realistas para permitir que objetos reais e virtuais interajam de forma plausível."³

De todos os sistemas apresentados neste capítulo, este é o mais parecido com o desenvolvido aqui. Entre outras formas de interação, ele usa uma mini-empilhadeira real controlada por controle remoto para movimentar objetos virtuais. Em cima da empilhadeira há um marcador para detectar sua posição. Ao movimentar as pás da empilhadeira real, é possível levantar uma caixa virtual na cena.

2.8 Considerações Finais

Há diversos sistemas de realidade aumentada com aplicações diversas e que encontram desafios importantes para sua comercialização efetiva. Apesar do sistema desenvolvido aqui não ser inovador, ele mostrou ser um exercício muito proveitoso para o autor, ensinando vários conceitos tanto de realidade aumentada como de computação gráfica e geometria computacional.

De fato, o uso de realidade aumentada pode ser muito útil para aumentar a imersão do usuário. Somente os usos apresentados aqui incluem jogos eletrônicos, tele-conferência e jogos de tabuleiro. É certo que há inúmeras outras aplicações.

No próximo capítulo é explicado o funcionamento detalhado de cada biblioteca usada e como elas foram usadas no sistema de criação de cenas interativas.

³Original: "AR games require sophisticated physics simulation, as the purpose of any AR application is to create the illusion that real and virtual objects coexist. This requires the addition of realistic physical motion to allow real and virtual objects to interact together in a way that appears plausible.", Fonte: [6].

3 *Desenvolvimento do Trabalho*

3.1 Considerações Iniciais

Primeiramente o autor precisou se familiarizar com as três bibliotecas usadas: Panda3D, ARToolKit e Bullet, as três possuem licenças permissivas (*free*) para uso. Após aprender o suficiente para criar uma cena simples usando Panda3D, um demo foi criado usando ARToolKit e Panda3D (sem a física) como prova de conceito. Após fazer isto com sucesso o autor criou o sistema de posicionamento de objetos, com a geração procedural dos objetos baseados em parâmetros escolhidos pelo usuário e o sistema de anexação de objetos à cena. Só então a física foi adicionada, para cada tipo de objeto foi preciso gerar proceduralmente um corpo rígido (veja seção 3.2.3) usando a biblioteca Bullet. Com sistema básico completo, alguns novos refinamentos foram feitos (em especial a criação de uma interface para ser usada com o mouse) e novos objetos foram adicionados.

No resto deste capítulo é descrito o sistema utilizado para o desenvolvimento, o funcionamento de cada biblioteca usada, como o sistema funciona internamente e o processo de criação de objetos e corpos rígidos proceduralmente.

3.2 Projeto

Para a criação do sistema foi usado um equipamento com Ubuntu 11.10, 32bits e a versão 1.8.0 da biblioteca Panda3D. Panda3D vem com as bibliotecas ARToolKit e Bullet já integradas em sua *API*. A câmera usada foi uma câmera USB A4Tech PK 810-G, mas qualquer câmera USB pode ser usada contanto que seja reconhecida pelo sistema operacional usado. Da mesma forma qualquer distribuição Linux deve ser capaz de ser usada com o sistema.

3.3 Instalação

No apêndice A pode ser encontrado a localização do código fonte criado assim como os outros arquivos necessários para executar o sistema. Para instalar o sistema basta ter uma distribuição Linux instalada e instalar a biblioteca Panda3D 1.8.0, esta versão já vem com as bibliotecas ARToolkit e Bullet. É necessário ter uma câmera, de preferência uma não embutida na tela (como a maioria dos notebooks possui) e imprimir os dois padrões (traço e kanji) usados. O marcador kanji (Figura 1.1) deve ser impresso com 5cm de lado e o padrão traço (Figura 1.2) com 2,5cm de lado. Para iniciar o sistema basta baixar todos os arquivos executar:

```
python main.py
```

Na pasta em que eles se localizam.

Após executar esta linha o sistema vai imprimir uma lista de todas as opções de câmeras presentes no sistema, basta digitar o número da desejada (melhores resultados são obtidos ao escolher uma opção com 640x480 de resolução e 30Hz de taxa de atualização). Boas condições de luz ajudam na detecção dos padrões.

3.4 Bibliotecas

Para criação do sistema foram usadas 3 bibliotecas: ARToolkit, Panda3D e Bullet. Nesta seção são explicados o funcionamento de cada uma delas detalhadamente.

3.4.1 ARToolkit

ARToolkit é uma biblioteca originalmente desenvolvida por Hirokazu Kato e licenciada via GPL. Ela provê detecção da posição e orientação de marcadores em relação à câmera, assim como treinamento de novos marcadores e calibração de câmera. Neste sistema foram usados os marcadores padrões providos pela biblioteca e não notou-se a necessidade de calibramento específico para a câmera usada.

A Figura 3.1 mostra o funcionamento da biblioteca. Primeiro a imagem é transformada em preto e branco e binarizada, em seguida usando técnicas de visão computacional as bordas do marcador são detectadas. Como o marcador é quadrado é possível extrair a posição e a orientação do marcador em relação à câmera a partir desta borda. Em seguida o símbolo dentro do marcador é comparado com os símbolos usados do sistema, finalmente, a geometria apropriada é aplicada ao centro do marcador.

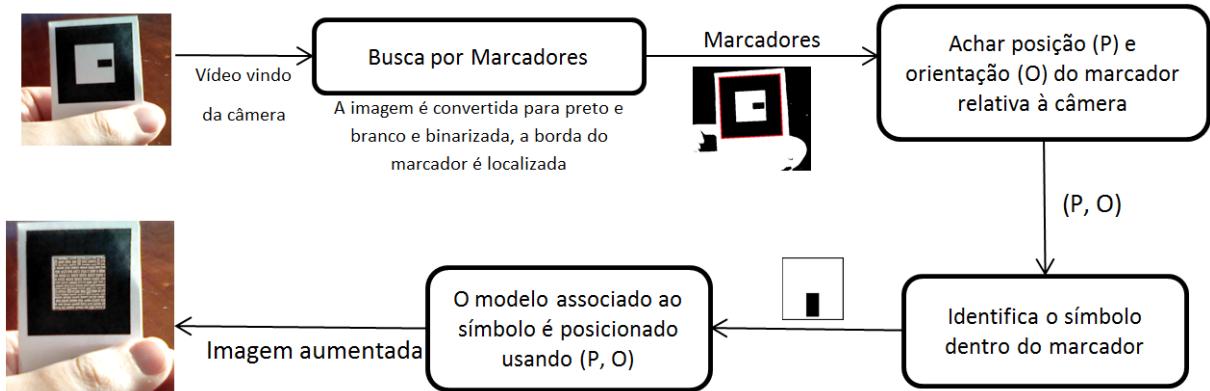


Figura 3.1: Funcionamento da biblioteca ARToolKit

3.4.2 Panda3D[7]

Panda3D na verdade é mais que uma biblioteca, é uma *engine* completa para criação de jogos, aplicações 3D e simulações. Ela suporta programação em python e em C++. Panda3D também inclui várias bibliotecas adicionais que proporcionam suporte à Som (através das bibliotecas FMOD, OpenAL ou Miles), Física (através da Bullet, ODE ou PhysX) e Realidade Aumentada (através da ARToolKit) entre outras coisas. A licença usada pela Panda3D é uma versão modificada da licença BSD e portanto Open Source. Algumas bibliotecas de terceiros contidas no Panda3D (por exemplo ARToolKit, Bullet e FMOD) possuem outras licenças, as restrições destas bibliotecas se aplicam somente quando elas são utilizadas. Todas as bibliotecas contidas no Panda3D são livres para ser usada em aplicações não comerciais.

Quanto ao funcionamento da parte de posicionamento (a parte mais importante para este sistema de realidade aumentada), a biblioteca possui um sistema de árvore de cena. Cada elemento desta árvore é um PandaNode. Esta árvore possui um PandaNode raiz chamado *render*. Um PandaNode possui vários atributos, entre eles posição euclidiana (X,Y,Z) e orientação (H,P,R). (H,P,R) vem de *Heading*, *Pitch* e *Roll* e controlam a rotação de um objeto ao redor dos três eixos coordenados de uma geometria 3D. A Figura 3.2 exemplifica este sistema de coordenadas.

Um PandaNode também pode possuir um (ou mais) GeomNodes filhos, um GeomNode contém qualquer tipo de geometria renderizável. Um GeomNode só é renderizado caso esteja presente na árvore de *render*. Qualquer filho de um PandaNode assume o sistema de coordenadas do pai. Por exemplo se temos dois PandaNodes, PandaNode1 e PandaNode2 e o PandaNode2 é filho de PandaNode1 que por sua vez é filho de *render*. Se a posição de PandaNode1 é (5,0,0) e a de PandaNode2 é (0,3,0) temos que a posição de PandaNode2 em relação ao *render* será (5,3,0). Este raciocínio funciona de forma análoga para a orientação dos nós.

Como exemplo Figura 3.3(a) é uma cena criada, ela possui 5 PandaNodes e 3 GeomNodes.

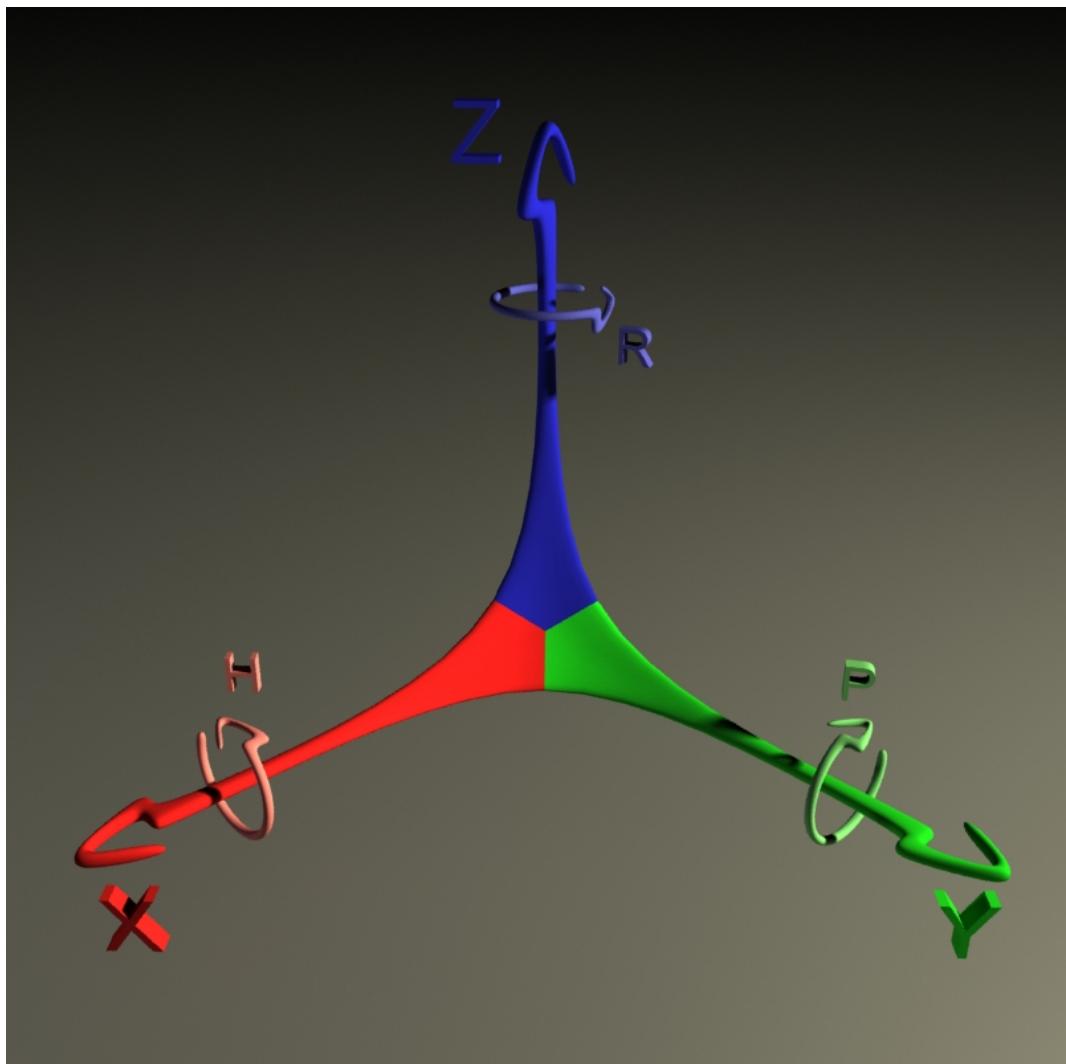


Figura 3.2: Sistema de coordenadas do Panda3D.

A Figura 3.3(b) é a árvore desta cena. É notável que a posição do óculos e do boné é relativa ao pai (panda1), sendo muito mais simples o posicionamento destes elementos em relação ao pai do que em relação ao centro da cena. Quando o panda1 é movido estes dois elementos não precisam ser reposicionados pelo programador manualmente

Este sistema de árvore de cena foi muito útil para o desenvolvimento do projeto. Ao anexar um objeto na cena usando o marcador traço, o PandaNode deste objeto passa a ser filho do PandaNode que fica preso à posição do marcador kanji, este processo é explicado com mais detalhes na seção 3.4.

3.4.3 Bullet[8]

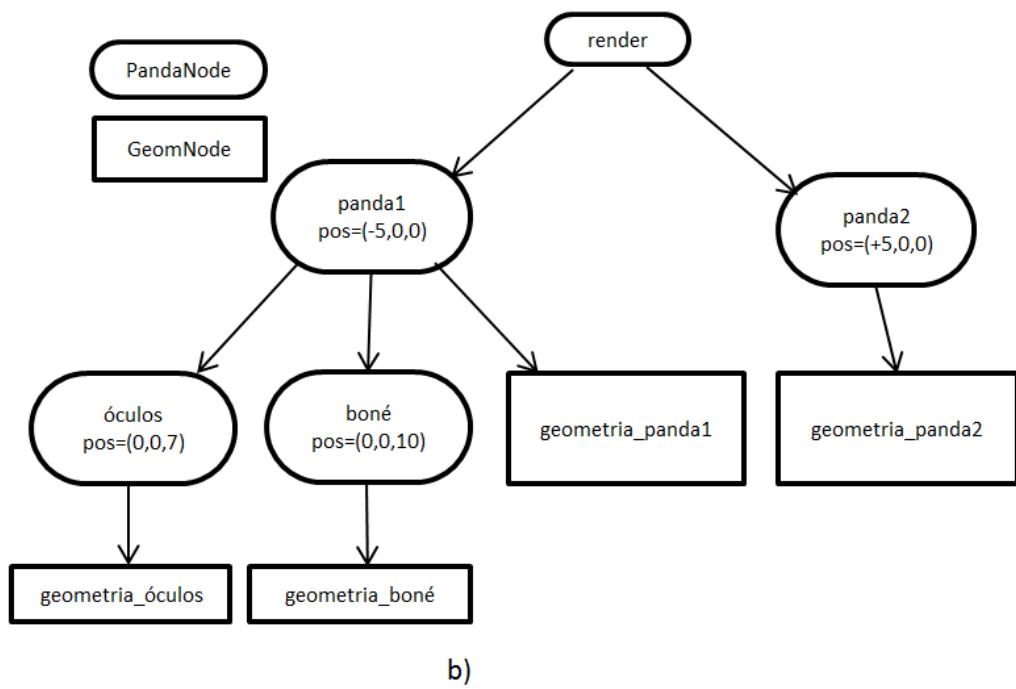
Bullet é uma biblioteca que proporciona simulações de física. Já foi usada com sucesso em vários jogos comerciais como Grand Theft Auto IV e filmes como 2012. Ela proporciona simulações de corpo rígido e corpo mole (um corpo rígido não pode ser deformado, um corpo mole pode). Estes corpos possuem vários parâmetros como massa e fricção. No sistema apresentado aqui foram usados somente corpos rígidos e apenas a massa é alterada, os outros parâmetros são mantidos em seus valores padrões. A licença usada pela Bullet é a zlib.

A integração da Bullet com o Panda3D é notavelmente simples. A Bullet define uma entidade chamada *World*, esta entidade pode possuir vários corpos rígidos ou moles anexadas à ela (*World* não é uma árvore, apenas uma lista de corpos). Cada corpo rígido pode referenciar um PandaNode de modo que ele fica em controle da posição e orientação deste PandaNode (e portanto de todos os seus filhos também). Há uma notável exceção à esta regra, se o corpo rígido possuir massa zero a posição dele será determinada pelo PandaNode que ele referencia e não o contrário.

A Figura 3.4 demonstra como um cubo formado por 6 PandaNodes (cada um contendo um dos lados do cubo) é referenciado por um corpo rígido de formato cúbico. Para facilitar a explicação, daqui em diante será usado o nome que o Bullet usa, *RigidBody*. Quando o *RigidBody* do cubo é afetado por uma força (como a gravidade, ou uma colisão com outro *RigidBody*) ele muda a posição e orientação do PandaNode *physicsCubo* (e de todos os seus filhos e portanto de toda a sua geometria).



a)



b)

Figura 3.3: Funcionamento da biblioteca Panda3D. Filhos assumem o sistema de coordenadas do pai.

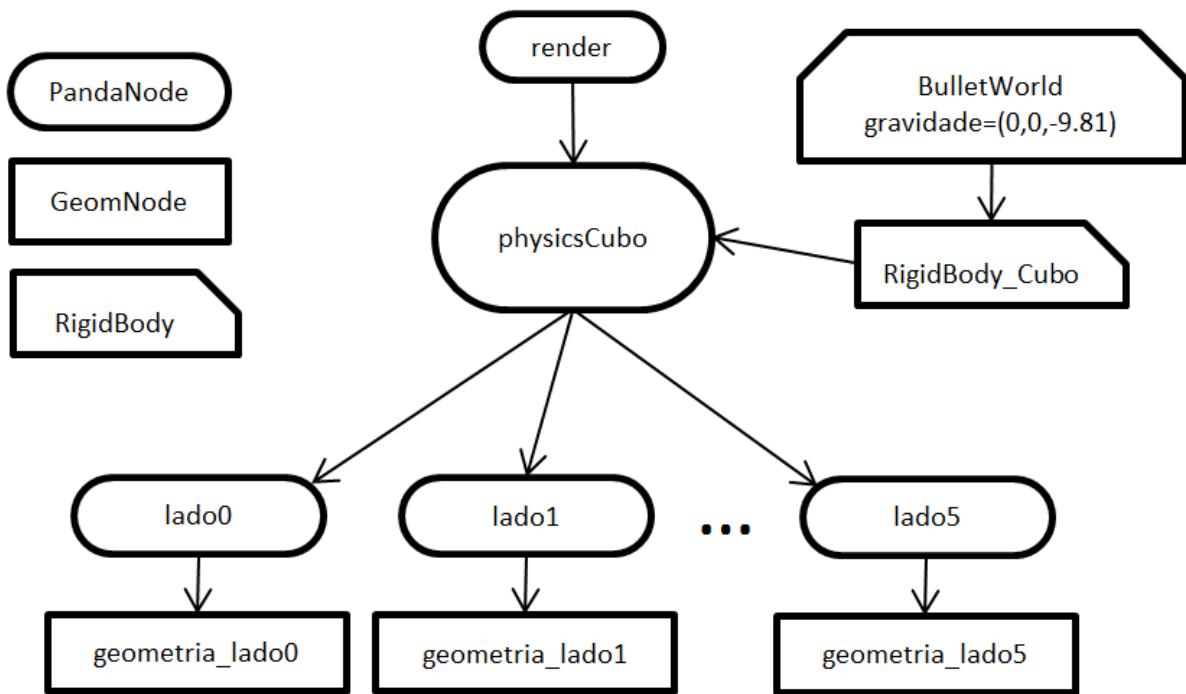


Figura 3.4: Funcionamento da biblioteca Bullet. Um RigidBody referencia um PandaNode assumindo controle de sua posição e orientação na cena.

3.5 Cena

Primeiramente a imagem obtida pela câmera é colocada como fundo da cena. Em seguida objetos são renderizados em cima da imagem obtida pela câmera gerando assim a cena. Controladores visuais também são mostrados para o usuário poder escolher e mudar os parâmetros do objeto preso ao marcador traço.

Para discutir a criação da cena é preciso ter conhecimento do funcionamento das bibliotecas usadas conforme explicado na seção 3.3. Há dois elementos principais sempre presentes na cena, dois PandaNodes aqui chamados de centroNP e objetoNP. Quando um marcador é detectado pela ARToolKit ele fica preso à posição e orientação fornecida pela biblioteca. O centroNP fica preso ao marcador kanji e o objetoNP fica preso ao marcador traço.

O PandaNode centroNP é referenciado por um RigidBody de formato plano infinito, este plano possui massa zero e portanto é a posição de centroNP que controla a posição do plano e não o contrário. Por sua vez centroNP tem sua posição e orientação definida pela posição do marcador kanji na imagem obtida pela webcam, assim o chão da cena sempre será o plano paralelo ao qual o marcador está (se o marcador estiver em cima de uma mesa, o chão será paralelo à mesa).

A gravidade da cena é feita de modo que ela sempre fique perpendicular ao marcador kanji

e possui módulo igual à 9.81 m/(s*s) (aproximadamente a gravidade da Terra). Se o marcador estiver apoiado em um plano paralelo ao chão da cena real (em cima de uma mesa, por exemplo) a gravidade da cena aumentada será igual à gravidade da cena real. Como a presença do marcador kanji também cria um chão, objetos virtuais podem colidir este chão, fornecendo assim a ilusão que os objetos da cena aumentada estão sujeitos às mesmas forças que afetam objetos reais.

O PandaNode `objetoNP` é o PandaNode preso ao marcador traço, à ele são anexados outros PandaNodes. Estes PandaNodes que possuem a geometria e os `RigidBody` dos objetos, apenas um objeto pode estar como filho de `objetoNP` de cada vez. A criação dos objetos é descrita na seção 3.5. Apesar de ser possível alterar a massa do objeto, enquanto um objeto está como filho de `objetoNP` ele possui massa zero (sobrescrevendo o valor de massa atribuído pelo usuário), fazendo com que ele sempre fique em cima do marcador. Isto é necessário para o usuário poder usar o objeto para interagir com a cena enquanto a física está em funcionamento. Se o objeto tivesse massa ele iria simplesmente cair.

A Figura 3.5 mostra a árvore *render* da Panda3D e o *world* da Bullet com um cubo preso ao padrão traço numa cena sem nenhum objeto anexado à ela ainda. Ao anexar o cubo à cena o cubo passa a ser filho de `centerNP`, a sua posição e orientação é alterada para ficar na mesma posição em relação à câmera, ou seja, para o usuário o cubo não muda de lugar, mas internamente a sua posição e orientação são convertidas para o sistema de coordenadas de `centerNP`. A massa escolhida pelo usuário é atribuída ao cubo, deste modo o cubo pode ser afetado pela gravidade. Um novo cubo (`cubo2`) com massa zero é criado e anexado à `objetoNP`. A Figura 3.6 mostra a cena após a anexação descrita assumindo que o usuário tenha escolhido `massa=10` para o cubo.

De forma análoga podemos anexar outros objetos diferentes à cena. `centerNP` pode possuir inúmeros outros objetos anexados, mas `objetoNP` sempre possui apenas um.

3.6 Criação dos Objetos

Nesta seção será discutido o processo de criação dos objetos usados para criar as cenas. Os objetos são:

- 1) Plano: Um simples plano com uma textura de tijolo.
- 2) Cubo: Um cubo com cada lado de uma cor diferente.
- 3) Esfera: Uma esfera com uma textura com um *smiley*.

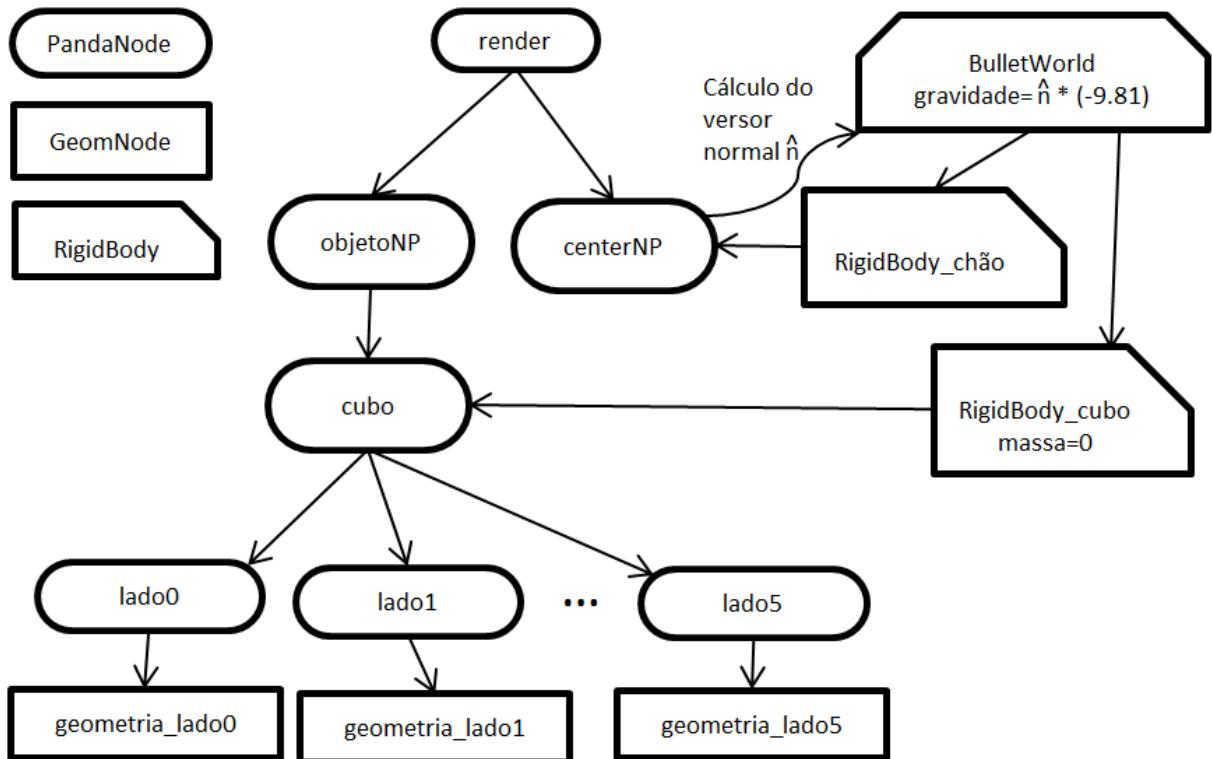


Figura 3.5: Árvore de uma cena sem nenhum objeto anexado ainda. Apenas um cubo presente preso ao marcador traço.

4) Estrada: Um plano com uma textura de estrada, o usuário pode escolher colocar um *guard-rail* ou não na estrada.

5) Curva de Estrada: Uma curva em 90 graus que usa a mesma textura de estrada, similarmente o usuário pode escolher colocar um *guard-rail* ou não na curva.

Destes objetos apenas a Esfera usa um modelo pré-feito e pré-texturizado. Todos os outros objetos são criados proceduralmente. A seguir será explicado o processo de criação de cada objeto e o mapeamento de textura usado. Lembrando que a direção Z é aquela que ”sai” do marcador (direção perpendicular ao marcador).

3.6.1 Criação de Retângulos Procedimentalmente

Todos os objetos que são criados proceduralmente são feitos de retângulos. O plano é apenas um retângulo, o cubo possui 6 retângulos e assim por diante. Para criar um retângulo é necessário ter dois pontos no espaço $P1=(x_1,y_1,z_1)$ e $P2=(x_2,y_2,z_2)$, tendo estes dois pontos é possível criar 4 vértices:

$$V1 = (x_1, y_1, z_1),$$

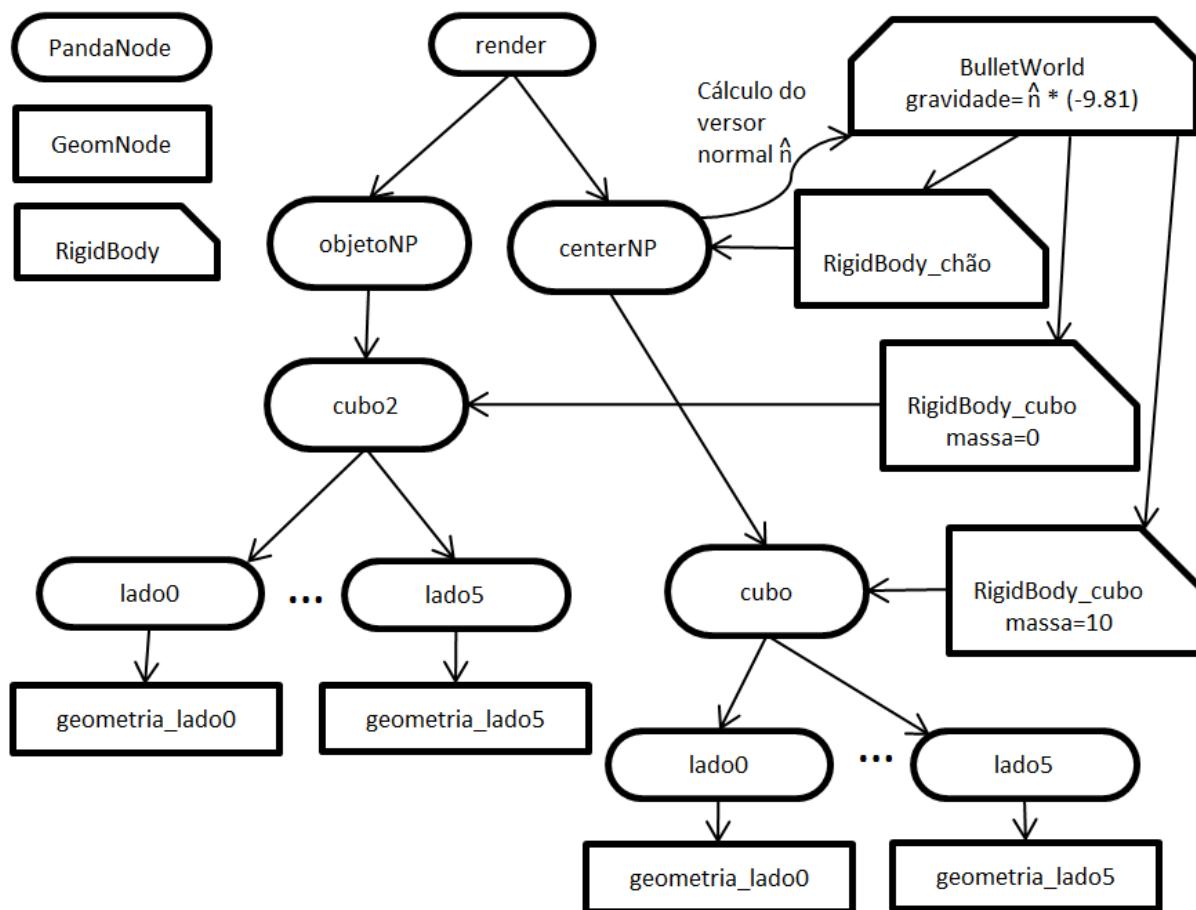


Figura 3.6: Árvore de Cena 1 após anexar o objeto à cena. Um novo cubo (cubo2) é criado.

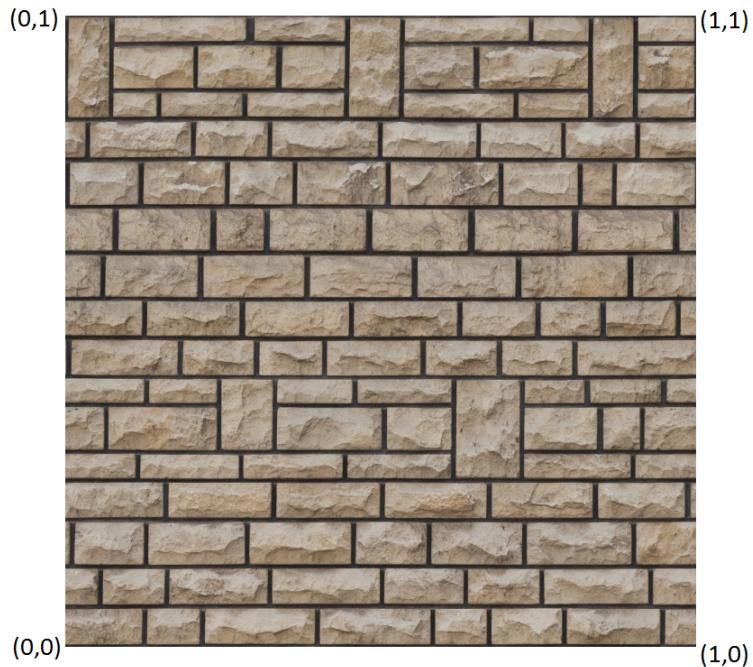


Figura 3.7: Uma textura quadrada de tijolo e sua indexação em (U, V).

$$V2 = (x_2, y_1, z_1),$$

$$V3 = (x_2, y_2, z_2),$$

$$V4 = (x_1, y_2, z_2)$$

Usando estes quatro vértices é criado um GeomNode com geometria de um retângulo. Cada objeto define estes pontos, P1 e P2, de forma diferente. O ponto (0,0,0) é o centro do objeto.

3.6.2 Mapeamento de Texturas

O método de mapeamento usado foi o método UV de mapeamento. No mapeamento UV a textura 2D é indexada em duas direções: U e V (não é usado X e Y pois estas duas letras já são usadas para o objeto 3D). U e V variam de 0 à 1 conforme mostrado na Figura 3.7.

Ao criar os objetos é definido um conjunto de coordenadas de textura, indicando como as texturas são mapeadas. Para facilitar o desenvolvimento todas as texturas são aplicadas *two-faced*, ou seja, elas são aplicadas aos dois lados dos polígonos.

3.6.3 Criação de Corpo Rígido

A biblioteca Bullet possui uma forma bastante útil para criação de corpos rígidos. É possível passar um GeomNode para a Bullet e obter uma Forma (de agora em diante chamada de Shape)

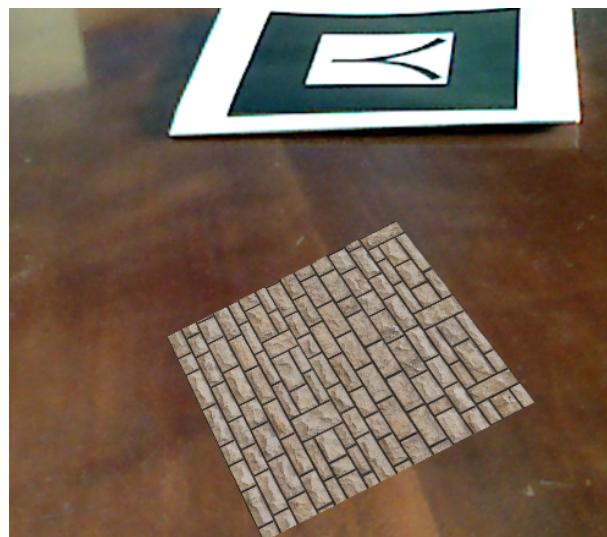


Figura 3.8: Um plano numa cena.

apropriada. Passando um GeomNode com um retângulo é obtido um Shape com formato de plano com as mesmas dimensões do GeomNode. Um RigidBody é um conjunto com um ou mais Shapes. O plano possui apenas um Shape. O cubo por outro lado possui 6, já que ele possui 6 retângulos. A união dos 6 Shapes gera o RigidBody do cubo.

Nas próximas seções é explicado a criação de cada objeto, para cada retângulo criado para confeccionar um objeto também é criada um Shape e um RigidBody com todas os Shapes é criado no final para o objeto poder ser afetado pela física de cena.

3.6.4 Plano

A Figura 3.8 mostra um plano anexado em uma cena. A textura usada no plano é a de tijolo (Figura 3.7), o usuário pode escolher o tamanho dos lados do plano na direção X e na direção Y, assim como a massa. De modo que:

tamX = tamanho na direção X,

tamY = tamanho na direção Y,

$P1 = (\text{tamX}/2, \text{tamY}/2, 0)$

$P2 = (-\text{tamX}/2, -\text{tamY}/2, 0)$

O mapeamento de textura é simples, ele apenas estica a textura para preencher o polígono totalmente.

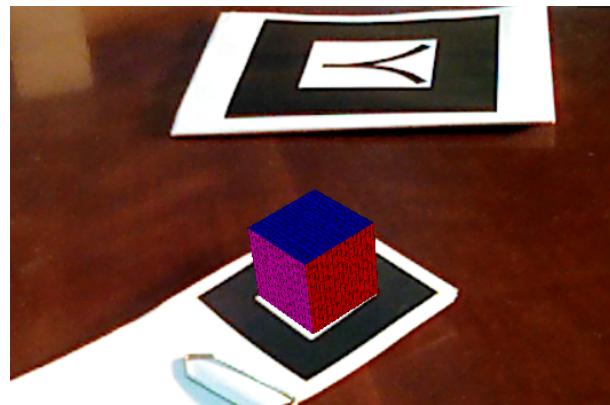


Figura 3.9: Um cubo numa cena.

3.6.5 Cubo

A Figura 3.9 mostra um cubo anexado em uma cena. A textura usada no cubo é a de tijolo (Figura 3.7), o usuário pode escolher o tamanho dos lados do cubo na direção X, Y e Z, assim como a massa. A cada lado do cubo foi atribuída uma cor diferente. O mapeamento de textura de cada um dos seis lados é análogo ao do Plano.

Lado 1:

$$P1 = (-\text{tamX}/2, -\text{tamY}/2, -\text{tamZ}/2), P2 = (\text{tamX}/2, -\text{tamY}/2, \text{tamZ}/2)$$

Lado 2:

$$P1 = (-\text{tamX}/2, \text{tamY}/2, -\text{tamZ}/2), P2 = (\text{tamX}/2, \text{tamY}/2, \text{tamZ}/2)$$

Lado 3:

$$P1 = (-\text{tamX}/2, \text{tamY}/2, \text{tamZ}/2), P2 = (\text{tamX}/2, -\text{tamY}/2, \text{tamZ}/2)$$

Lado 4:

$$P1 = (-\text{tamX}/2, \text{tamY}/2, -\text{tamZ}/2), P2 = (\text{tamX}/2, -\text{tamY}/2, -\text{tamZ}/2)$$

Lado 5:

$$P1 = (-\text{tamX}/2, -\text{tamY}/2, -\text{tamZ}/2), P2 = (-\text{tamX}/2, \text{tamY}/2, \text{tamZ}/2)$$

Lado 6:

$$P1 = (\text{tamX}/2, -\text{tamY}/2, -\text{tamZ}/2), P2 = (\text{tamX}/2, \text{tamY}/2, \text{tamZ}/2)$$

Apesar de aqui ser chamado de Cubo, na verdade o objeto não precisa ter todos os lados de mesmo tamanho.

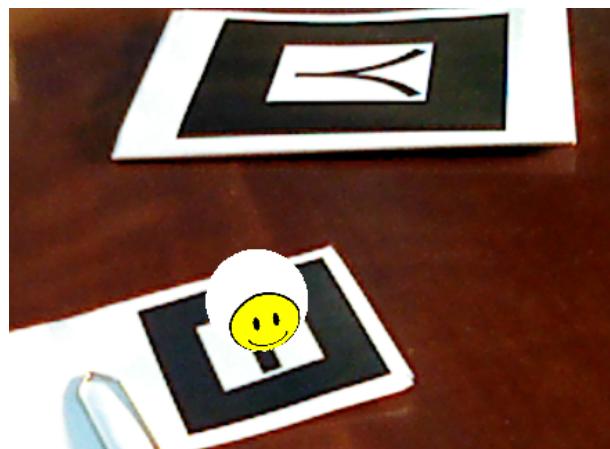


Figura 3.10: Um esfera numa cena.

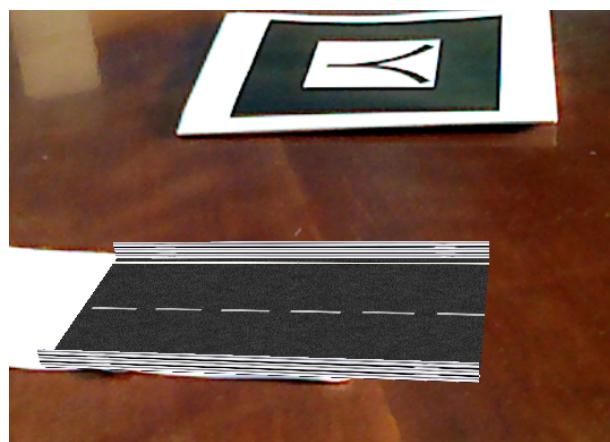


Figura 3.11: Um estrada numa cena.

3.6.6 Esfera

A Figura 3.10 mostra uma esfera anexada em uma cena. A esfera é pré-modelada e pré-texturizada. O usuário pode apenas escolher o raio da esfera. Para criar o RigidBody ao invés de passar o GeomNode, é usado uma primitiva Shape de formato esférico que a Bullet disponibiliza. Este Shape possui um raio igual ao do modelo.

3.6.7 Estrada

A Figura 3.11 mostra uma estrada com *guard-rail*. O usuário pode escolher a presença ou não do *guard-rail*, assim como o tamanho na direção X e na direção Y do objeto. A estrada e os *guard-rails* sempre possuem massa zero para eles ficarem fixos na cena. A criação da geometria da estrada sem o *guard-rail* é análoga ao plano, mas o mapeamento de texturas é diferente. Como a estrada geralmente possui um comprimento (direção X) muito maior do que sua largura (direção Y), a textura é esticada apenas na direção Y. Na direção X a textura se

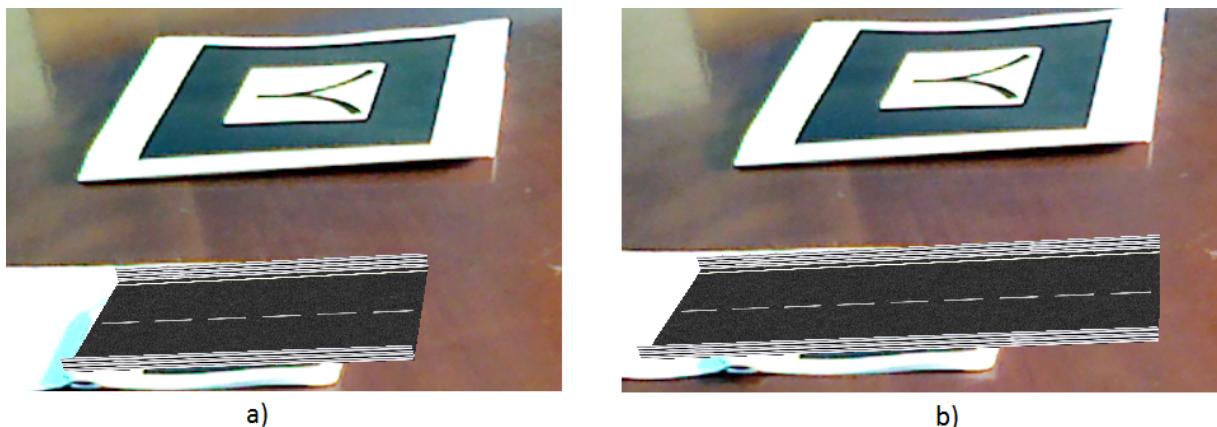


Figura 3.12: a) Uma estrada com tamY=1 e tamX=2, b) Uma estrada com tamY=1 e tamX=5.

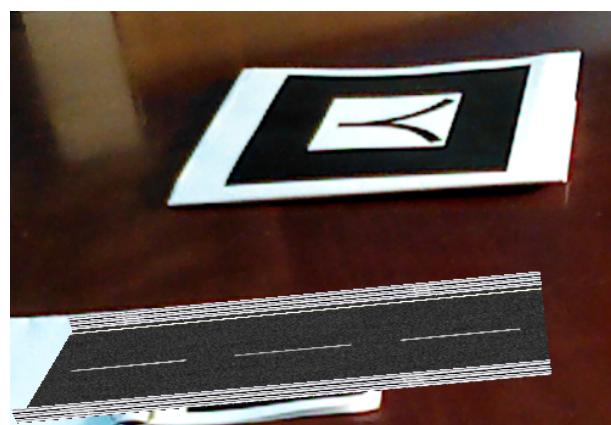


Figura 3.13: Uma estrada criada com $L=1$, com $\text{tamY}=1$ e $\text{tamX}=7$.

repete usando a razão:

$$L = \frac{tamX}{tamY}$$

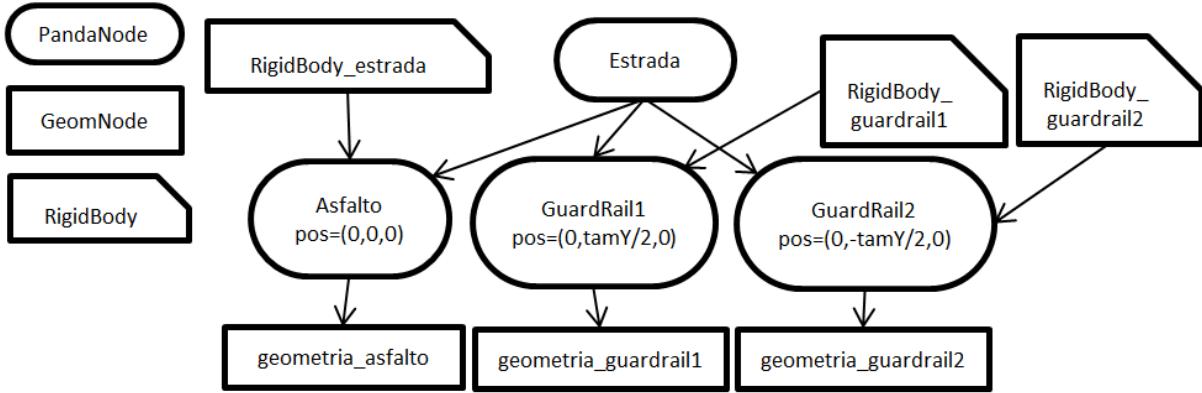
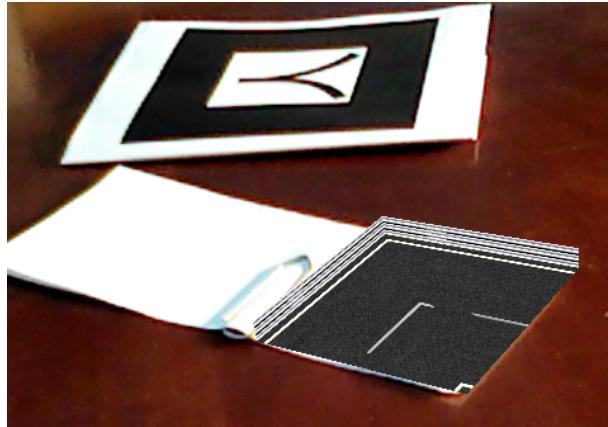
Esta razão L (geralmente) é maior que 1 e ela é usada para indicar quantas vezes a textura se repete na direção X. A textura é indexada de 0 à 1, se L=2.5 a textura vai se repetir duas vezes e meia na direção X conforme exemplificado na Figura 3.12. Esta razão é necessária pois a textura é esticada na direção Y, se L=1 fosse usado a textura iria ficar demasiada esticada na direção X conforme mostrado na Figura 3.13.

Como dito anteriormente, o *guard-rail* é opcional. A Figura 3.14 mostra a árvore da estrada com o *guard-rail* presente. Cada *guard-rail* é um quadrado criado usando os seguintes pontos:

guard-rail1:

$$P1=(tamX/2, 0, 0), P2=(-tamX/2, 0, tamY/10)$$

guard-rail2:

Figura 3.14: A árvore de uma estrada com *guard-rail*Figura 3.15: Curva de Estrada com *guard-rail*

$$P1=(\text{tamX}/2, 0, 0). P2=(-\text{tamX}/2, 0, \text{tamY}/10)$$

De modo que o comprimento do *guard-rail* é o mesmo da estrada e a altura do *guard-rail* é a largura da estrada dividido por 10. A posição dos *guard-rail* são $(0,\text{tamY}/2,0)$ e $(0,-\text{tamY}/2,0)$ pois eles estão na borda da estrada e não no centro. O mapeamento de textura é igual ao da estrada, repetindo a textura na direção X. Numa cena normal o PandaNode Estrada seria um filho direto de objetoNP ou centroNP.

3.6.8 Curva de Estrada

A Figura 3.15 mostra uma curva de estrada com *guard-rail*. Este objeto foi feito para ser usado em conjunto com o objeto Estrada para montar um circuito. A curva de estrada é sempre quadrada, tendo somente um parâmetro tamanho (tam). Assim como a estrada normal, é possível retirar o *guard-rail*. O que a curva difere da estrada normal é a posição dos *guard-rails* e o mapeamento de textura. A textura usada é a mesma da textura da estrada normal, o mapeamento da textura que difere.

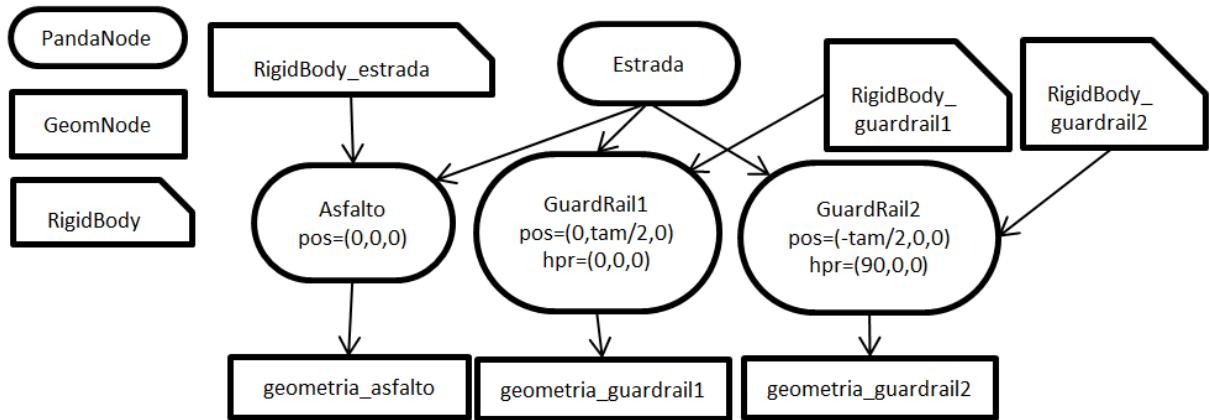


Figura 3.16: Árvore de uma Curva de Estrada com *guard-rail*

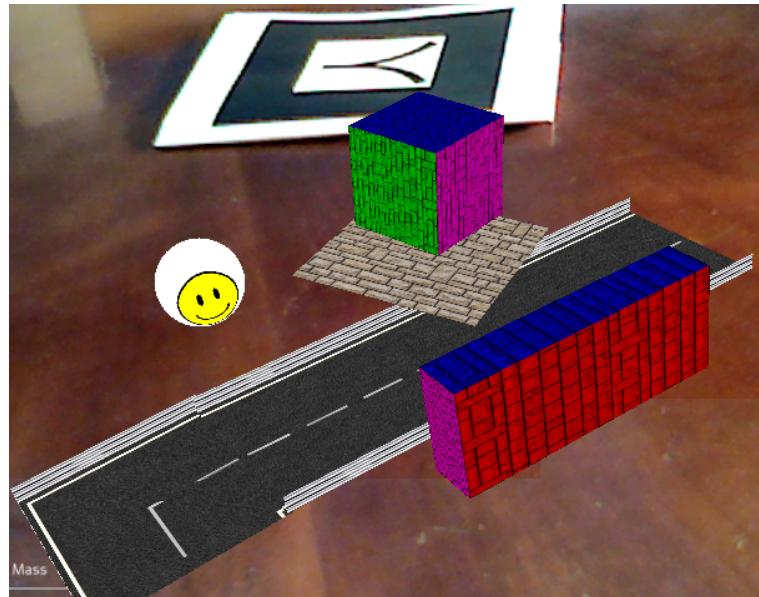


Figura 3.17: Uma cena com vários objetos.

A Figura 3.16 mostra a árvore da curva de estrada. Cada *guard-rail* é um quadrado criado da mesma forma da estrada normal, mas a posição do GuardRail2 é ($-tam/2,0,0$) e sua orientação é (90,0,0), ou seja, o *guard-rail* é girado em 90 graus para ficar perpendicular ao primeiro *guard-rail*.

3.7 Resultados Obtidos

Os resultados mostram que é possível criar um sistema de realidade aumentada com simulação de física que funciona de modo satisfatório. Os objetos são criados de forma adequada e o sistema de anexação à cena torna simples o posicionamento dos objetos. A física afeta os objetos da forma esperada. A detecção dos padrões funciona bem, mas dependendo das



Figura 3.18: Controladores para alterar os parâmetros de um objeto. Apenas os controladores apropriados ao objeto preso ao marcador traço são mostrados.

condições de iluminação e do ângulo do marcador em relação à câmera a detecção pode falhar. A Figura 3.17 mostra uma cena com vários objetos presentes ao mesmo tempo.

3.7.1 Funcionamento

No sistema a física pode estar ligada ou desligada, em geral é melhor posicionar os objetos com a física desligada para não haver colisões acidentais. A física pode ser ligada ou desligada usando a tecla enter do teclado.

É desejável manter a interação usando mouse ou teclado mínima, mas por praticidade o tamanho dos objetos, a sua massa, qual objeto está preso no marcador e a ação de ligar ou desligar a física são interações feitas por mouse e teclado. A Figura 3.18 mostra os controladores presentes para alterar os objetos usando o mouse.

No sistema é possível anexar um objeto preso ao marcador traço ao marcador kanji usando a tecla espaço do teclado. Ao fazer isto um novo objeto igual é criado e preso no padrão traço e o primeiro objeto é desanexado do padrão traço e passa a ser anexado ao padrão kanji, mas para o usuário o objeto não muda de lugar ao ser anexado. Desta maneira é possível posicionar inúmeros objetos na cena.

A gravidade usada na simulação da física da cena é sempre perpendicular ao padrão Kanji, então é ideal mantê-lo no canto da cena e imóvel enquanto a física está ativada. Desta forma marcador Kanji funciona como um ponto de referência para a cena de modo que a câmera não precisa ficar fixa em um ponto.

Ao ativar a física todos objetos que possuem massa diferente de zero irão ser afetados pela gravidade e cairão em direção ao chão mencionado anteriormente, colidindo com qualquer outro objeto no caminho. É possível atribuir massa 0 aos objetos de forma que eles não são afetados pela gravidade e colisões, mas outros objetos ainda podem colidir com eles (de fato, o objeto

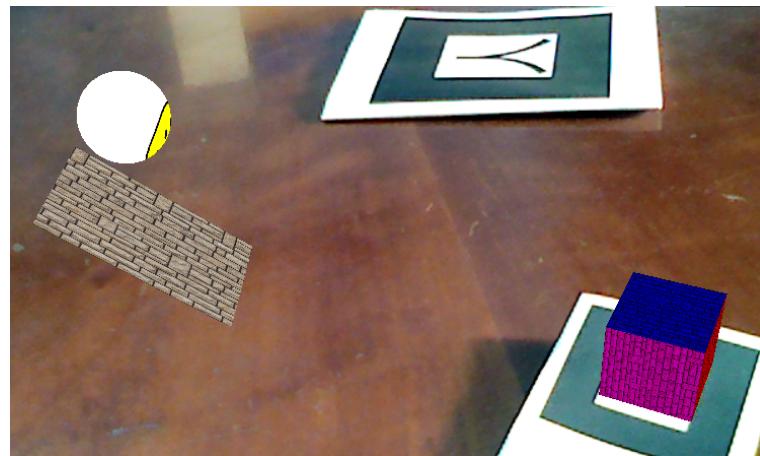


Figura 3.19: Cena criada usando o sistema.

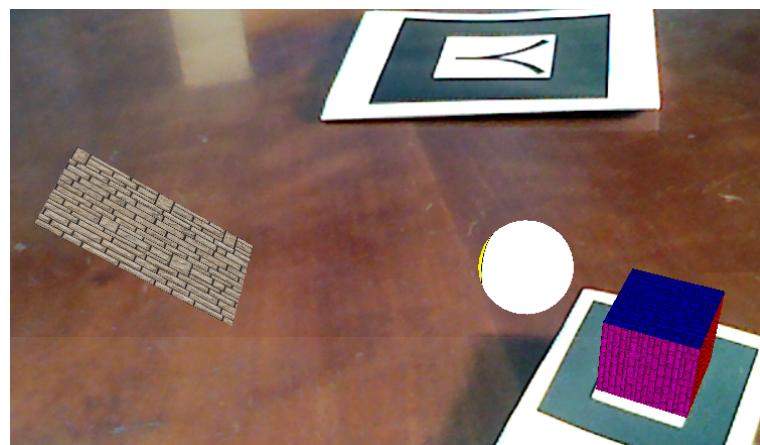


Figura 3.20: A mesma cena da Figura 1.4 alguns segundos depois de ativar a física.

preso ao padrão traço possui massa 0 até ser anexado à cena).

A Figura 3.19 mostra uma cena criada, nela o plano mais a esquerda possui massa zero, o cubo da direita está preso ao padrão traço. A esfera possui massa diferente de zero. A Figura 3.20 mostra a mesma cena alguns segundos depois. Note que a esfera colidiu e rolou pelo plano antes de cair no chão e continuar rolando.

3.8 Dificuldades

A principal dificuldade encontrada foi o posicionamento dos objetos em cima dos padrões. Originalmente o autor estava desenvolvendo o sistema no Windows 7, mas a implementação da captura de vídeo da câmera no Panda3D possuía um bug que fazia os modelos ficarem deslocados do centro do marcador. A Figura 3.21 mostra este deslocamento.

O autor perdeu inúmeras horas tentando descobrir a causa do problema. Eventualmente o

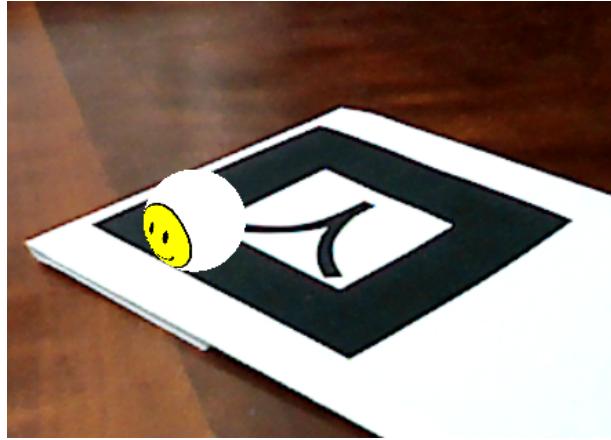


Figura 3.21: Bug presente no Panda3D no Windows, o modelo não fica exatamente em cima do centro marcador.

autor decidiu testar este posicionamento no Linux e descobriu o bug. Todo o desenvolvimento foi então transferido para um ambiente Linux, o que consumiu um tempo considerável já que o computador usado pelo autor possui uma placa de vídeo da nVidia com a tecnologia Optimus. Esta tecnologia não é suportada pelos drivers da nVidia no Linux, foi necessário instalar um software chamado Bumblebee que adiciona o suporte à Optimus. Outro problema de transferir o desenvolvimento para o Linux foi a falta de suporte para a câmera usada originalmente, foi preciso comprar uma nova câmera. Nesta nova câmera não é possível desligar a correção de brilho no Linux o que dificulta o reconhecimento dos padrões.

Outra grande dificuldade foi a familiarização do autor com as bibliotecas usadas. A integração da física da Bullet com o posicionamento dos modelos pela Panda3D não foi exatamente natural. Toda vez que um objeto é anexado à cena, o RigidBody deste objeto precisa ser atribuído a sua massa e anexado à *world*. Cuidar da árvore *render* e da lista de RigidBodys *world* é consideravelmente trabalhoso conforme visto pelas várias árvores descritas nas seções anteriores.

A criação procedural dos objetos foi consideravelmente trabalhosa, mas foi necessária para permitir que o usuário pudesse mudar o tamanho dos objetos de forma satisfatória. Os *guard-rails* em especial foram os mais trabalhosos. Em comparação com a esfera que é pré-modelada e pré-texturizada, o objeto mais simples que é o plano já é consideravelmente mais trabalhoso de ser criado.

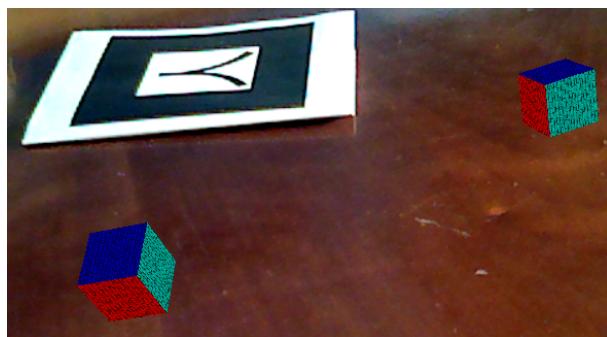


Figura 3.22: Falta percepção de profundidade dos objetos. O objeto da esquerda está muito mais próximo que o da direita.

3.9 Limitações

A principal limitação do sistema é que os marcadores precisam estar completamente dentro da visão da câmera, caso contrário eles não são detectados. É comum o usuário ao posicionar um objeto, bloquear a visão do marcador kanji com o marcador traço. Os dedos do usuário também não podem estar em cima do marcador.

Outra limitação é que a quantidade de quadros por segundo fica sincronizada com a taxa de atualização da câmera (geralmente 30 Hz). Este é um problema da implementação da captura de vídeo da Panda3D no Linux.

Como o sistema usa visão indireta, o usuário pode ficar desnorteado pois um movimento do marcador para a esquerda pode fazer o objeto se deslocar para a direita na tela. Para diminuir este problema é recomendado que a câmera fique posicionada à frente do usuário e não olhando na direção do usuário.

A falta percepção de profundidade dos objetos é um problema constante, como é possível ver na Figura 3.22. O objeto da esquerda está muito mais próximo que o da direita, mas eles parecem estar na mesma distância. Na verdade o objeto da direita é maior, mas está mais afastado.

3.10 Considerações Finais

O sistema mostrou ser muito mais complexo do que o esperado inicialmente. Mesmo com a ajuda das bibliotecas usadas, o desafio foi grande. Na opinião do autor, o resultado final ficou satisfatório, mas a limitação de falta de percepção de profundidade limita severamente um uso prático do sistema. De fato, sistemas deste tipo se beneficiam grandemente ao usar visão direta e estereoscópica, pois a imersão e maior e a falta de percepção de profundidade é reduzida.

No próximo capítulo é discutido as contribuições do sistema criado para o autor e para a comunidade acadêmica, assim como as considerações sobre o curso graduação.

4 Conclusão

4.1 Contribuições

O autor acredita que o trabalho é relevante pois tais sistemas de realidade aumentada estão se tornando cada vez mais viáveis, mas a pesquisa e o desenvolvimento na área ainda são relativamente pequenos.

Para o autor a criação do sistema provou ser uma ótima experiência, acrescentando muito conhecimento nas áreas de Computação Gráfica e Interação Humano-Computador.

4.2 Trabalhos Futuros

Uma possibilidade de expansão do trabalho seria adicionar a capacidade de controlar um carro que conseguiria andar sobre as estradas e interagir com os objetos da cena.

Outra possibilidade seria adicionar suporte para um sistema de visão direta e estereoscópica (um *head-mounted display* por exemplo), eliminando a falta de percepção de profundidade e de desnorteamento do usuário.

Seria também interessante adicionar suporte a multi-marcadores, entretanto o uso de multi-marcadores diminuiria a imersão do usuário pois o chão da cena sempre seria um papel branco com vários símbolos abstratos em volta. Outra possibilidade seria um sistema que analisasse a imagem da câmera a procura de pontos de referência na própria imagem. Estes pontos substituiriam os marcadores, mas a implementação seria extremamente mais complexa na área de visão computacional.

Mais interessante ainda seria o suporte para *smartphones*. A ubiquidade de *smartphones* e o fato de um sistema que usa a câmera de um *smartphone* ser um sistema de visão direta, tornaria o sistema muito mais viável. Por outro lado a biblioteca Panda3D não suporta nenhum sistema operacional de dispositivos móveis ainda, necessitando o sistema ser completamente reescrito.

4.3 Considerações sobre o Curso de Graduação

O curso de bacharelado em Ciências da Computação provou ser uma experiência valiosa para o autor, o conhecimento técnico e teórico obtido com certeza provará ser extremamente valioso, mas há muito espaço para melhorar o curso.

Primeiramente o curso é focado de mais em matérias de matemática, mas a nova grade que entrou em vigor em 2012 diminuiu grandemente estes requisitos. Infelizmente o autor não teve oportunidade de estudar nesta nova grade. Idealmente também teríamos professores da Computação ensinando matérias de matemática e estatística para os alunos focando nas áreas interessantes para alunos da Computação ao invés de professores de matemáticas mais preocupados que os alunos provem teoremas.

Em segundo lugar algumas das matérias da área de computação focam demais o lado teórico, entre elas estão: Inteligência Artificial, Sistemas Operacionais, Computação Gráfica, Redes de Computadores, Programação Matemática, Arquitetura de Computadores e Programação Concorrente. Todas estas matérias focaram demasiadamente a parte teórica em detrimento da parte prática, na opinião do autor todas estas matérias deveriam ser ensinadas com a mesma didática de Laboratório de Banco de Dados. Uma matéria teórica seguida por um aprofundamento com enfoque prático, mesmo que este aprofundamento fosse uma matéria optativa. Tal foco teórico faz com que a matéria seja muito fácil e não exige o suficiente do aluno.

Quanto aos professores, todos provaram possuir bastante conhecimento, mas muitos possuem problemas de didática. A recente obsessão da graduação com a presença dos alunos nas aulas pouco faz para ajudar. Os alunos não faltam nas aulas por serem preguiçosos, mas sim por causa do pouco proveito que eles tiram das aulas. Em geral é mais proveitoso para os alunos aprenderem as matérias por conta dos livros do que assistir uma aula que dura tempo demais com um professor com pouca didática numa matéria focada demais na parte teórica e que os alunos sabem que não vão usar em suas vidas profissionais.

Ao autor parece que as ementas das disciplinas foram criadas mais para ensinar o máximo de assuntos possíveis sem garantir que os alunos realmente entendam os assuntos. Para os alunos seria mais proveitoso aprender um número reduzido de assuntos, mas com mais profundidade e enfoque prático.

Por outro lado as matérias optativas são muito mais interessantes para os alunos, o interesse e aproveitamento aumenta muito quando o aluno realmente quer fazer a matéria. O enfoque mais prático também ajuda. Na opinião do autor todos os semestres a partir do se-

gundo deveriam ter créditos dedicados para disciplinas optativas. Várias disciplinas que hoje são obrigatórias deveriam ser optativas.

Referências Bibliográficas

- [1] KIRNER, C.; TORI, R. Fundamentos de Realidade Aumentada. Capítulo 2.
- [2] KATO, H.; BILLINGHURST, M. Marker tracking and hmd calibration for a video-based augmented reality conferencing system., In Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR 99), October 1999.
- [3] ODA, O.; LISTER, L.J.; WHITE, S.; FEINER, S. Developing an augmented reality racing game. Proc. of the 2nd Int. Conf. on INtelligent TEchnologies for interactive entertainment (INTETAIN'08). ICST, Brussels, Belgium, 2008.
- [4] LEE, J.Y.; SEO, D.W.; RHEE, G.W. Tangible authoring of 3D virtual scenes in dynamic augmented reality environment. Computers in Industry. v.62, n.1, 2011, pp.107-119
- [5] MOLLA, E.; LEPETIT, V. Augmented Reality for Board Games. Proc. of the IEEE Int. Symp. on Mixed and Augmented Reality (ISMAR 2010), pp.253-254
- [6] NAMEE, B.C.; BEANEY, D.; DONG, Q. Motion in Augmented Reality Games: An Engine for Creating Plausible Physical Interactions in Augmented Reality Games. International Journal of Computer Games Technology, v.2010, 2010.
- [7] GOSLIN, M. and MINE, M.R. The Panda3D Graphics Engine. Computer, v.37, n. 10, 2004, pp. 112-114
- [8] COUMANS, E. Bullet physics library. Disponível em: <http://bulletphysics.org/>. Acesso em: 30/05/2012.

Apêndice A - Código Fonte

O código fonte e arquivos necessários para executar o sistema podem ser obtidos em:

<http://code.google.com/p/cenas-realiade-aumentada/>

As texturas usadas no sistema foram obtidas do website CGTextures:

<http://www.cgtextures.com/>

Para instruções de instalação e uso veja seções 1.3 e 1.4.