

**Trabalho de Conclusão de Curso
Physimulation - Animação de Fenômenos
Físicos**

Rafael Issao Miyagawa Alberto Hideki Ueda

Orientadores: José Coelho de Pina Junior
 João Pedro Kerr Catunda

Novembro de 2012

The most exciting phrase to hear in science, the one that heralds new discoveries, is not 'Eureka!' but 'That's funny ...'

Isaac Asimov

Conteúdo

I Parte objetiva	4
1 Introdução	4
1.1 Física no BCC	4
1.2 Problemas ao simular interações físicas	5
1.3 Simulador	5
1.4 Estrutura da Monografia	5
2 Plataforma computacional	8
2.1 Esquema de dependências	8
2.2 Ruby	9
2.3 Simulação com Chipmunk	9
2.4 Animação com Gosu	9
2.5 Cenários com Glade	9
2.6 Exemplo	10
3 Discretização da simulação	11
3.1 Tempo de simulação	11
3.2 Tempo fixado	11
3.3 Tempo variável	11
4 Colisões	12
4.1 Algoritmos para detecção	12
4.2 Colisões no Chipmunk	12
4.3 Efeito de colisões	12
5 Atividades realizadas	13
5.1 Estudo inicial	13
5.2 Ambientes de demonstração	15
5.3 Criador de Cenários	16
6 Animações produzidas	18
6.1 Descrição	18
6.2 Visualização das animações	25
7 Physimulation	26
7.1 A ferramenta	26
7.2 Criando um cenário básico	26
7.3 Exemplos	27

7.4	Arquivos de configuração	31
7.5	Pontos de modificação	31
8	Introdução à computação com animações	32
8.1	Configuração	32
8.2	Angry Bixos	32
8.3	Apolo	32
9	Comentários finais	33
9.1	Trabalhos futuros	33
9.2	Desafios	33
9.3	Conclusão	33
10	Instruções de instalação da plataforma	34
11	Apêndice	35
II	Parte subjetiva	36
11.1	Desafios e frustrações encontrados	36
11.2	Disciplinas mais relevantes - Alberto Ueda	37
11.3	Disciplinas mais relevantes - Rafael Miyagawa	39
11.4	Estudos futuros	39

Parte I

Parte objetiva

1 Introdução

O foco deste projeto é atrair a atenção dos alunos do IME em relação às disciplinas de física ministradas no curso de bacharelado em Ciência da Computação.

1.1 Física no BCC

Embora seja comum encontrar alunos da computação que tenham interesse em tópicos relacionados a física clássica e moderna - Leis de Newton, Leis de Kepler, Teoria da Relatividade de Einstein, entre outros - há considerável resistência do corpo discente em relação a grade curricular do BCC incluir as disciplinas de física. (TODO citar questionário BCC) Isto não se aplica apenas às disciplinas de física, mas também a álgebra, cálculo, probabilidade e estatística (TODO referência).

Visando compreender melhor a situação e apresentar uma proposta de reformulação da grade ou das disciplinas, foi criado o Grupo de Apoio ao BCC (TODO nomes), uma iniciativa da Comissão de Coordenação do BCC (CoC). Após a aplicação de questionários e conversas nos Encontros BCC (TODO citar), um dos consensos de ambas as partes - professores e alunos - foi o de que seria interessante uma maior contextualização destas disciplinas para os alunos em termos de computação. No apêndice da seção 11 há uma cópia de um textos produzidos pelo CoC, com referências para o documento original.

Tal contextualização pode ser feita de várias formas: por parte dos professores das disciplinas - mostrando aplicações dos tópicos estudadas para computação, trazendo professores ou especialistas em sala de aula que possam transmitir experiências relacionadas ao conteúdo aprendido - ou por parte dos próprios alunos - reunindo-se para palestras extra-classe, desenvolvendo bibliotecas que motivem os demais alunos a utilizar, pesquisar e até mesmo programar baseando-se nos conceitos adquiridos.

Nesse sentido, o Physimulation é uma ferramenta para simulação e animação de fenômenos físicos e tem como um dos principais objetivos integrar o conhecimento adquirido em FAP-126 (Física I) com um ambiente de programação (TODO programa?).

Com este trabalho visamos contextualizar os assuntos abordados na disciplina com demonstrações de ambientes físicos, integrações com exercícios-programas (EP's) e a apresentação de desafios atuais de simulação no campo da física.

1.2 Problemas ao simular interações físicas

Existem muitos problemas ao tentar simular um ambiente físico com a computação. Temos o problema do tempo de simulação, que é o tempo que damos para os objetos físicos se interagirem e tomar o rumo necessário para refletir a realidade.

Existem alguns conceitos como *broad phase*, que é a fase em que os objetos são filtrados para realizarmos depois a *narrow phase* que é onde verificamos se aconteceu alguma colisão entre os objetos.

1.3 Simulador

Com o Physimulation, o aluno da disciplina de física poderá simular comportamentos físicos estudados em sala de aula, por meio de criação de cenários e objetos com diferentes propriedades configuráveis. Exemplos de propriedades são: massa, tamanho, posição, velocidade inicial, coeficiente de atrito e de elasticidade.

Desde rampas, pequenas esferas e plataformas fixas a planetas orbitando em torno de um objeto de grande massa, tanto os alunos quanto o professor poderão utilizar o simulador para combinar tais objetos e observar resultados calculados em exercícios. Além disso, estes cenários podem ser salvos no sistema e carregados posteriormente para nova observação.

1.4 Estrutura da Monografia

Esta monografia está estruturada da seguinte forma: Na seção 2 descrevemos os conceitos e tecnologias estudadas para a realização do trabalho. São apresentadas as bibliotecas utilizadas, o papel de cada uma em nosso projeto e como elas interagem entre si. Nas seções 3 e 4 explicamos dois tópicos comuns e importantes ao realizar simulações físicas: são eles, respectivamente, a discretização da simulação - definição de tempo de simulação, problema do tamanho do passo, granularidade - e o tratamento de colisões - algoritmos para filtragem de elementos, detecção de colisões e quais algoritmos são utilizados no Physimulation.

Já na seção 5 mostramos como foi o passo-a-passo de nosso trabalho, a metodologia utilizada e algumas dificuldades encontradas no caminho. A partir da seção 6, apresentamos um os resultados obtidos com o trabalho: primeiramente, os ambientes físicos de demonstração e as razões por ter começado o trabalho por eles. Logo depois, na seção 7 descrevemos o Physimulation e as etapas para a criação de cenários físicos. Como aplicação do projeto, apresentamos duas integrações com exercícios-programas de disciplinas de introdução a computação na seção 8.

Concluímos o trabalho na seção 9, além de citar possíveis trabalhos futuros e mostrar desafios atuais da computação em simulações físicas que ainda necessitam de estudo e aprimoramento. Ao final, há um roteiro para instalação do ambiente necessário para executar o Physimulation, na seção 10.

2 Plataforma computacional

2.1 Esquema de dependências

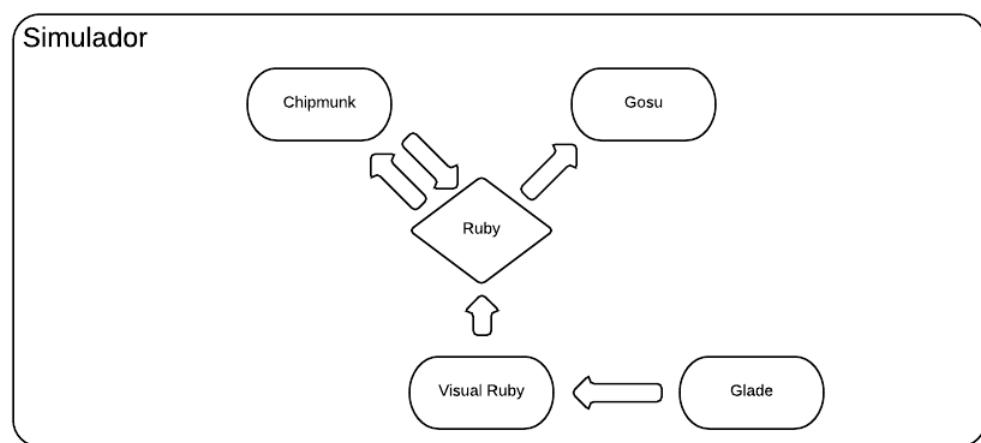


Figura 1: Dependências do Physimulation

2.2 Ruby

É a linguagem de programação utilizada no simulador. A sintaxe foi baseada no Pearl e Smalltalk e é uma linguagem interpretada com tipagem dinâmica e forte.

Algumas das características da linguagem são:

- Mostrar códigos e alguns itens interessantes do ruby

Explicar sobre gem.

2.3 Simulação com Chipmunk

Chipmunk é uma biblioteca física 2D escrita em C. Esta biblioteca permite criar objetos convexos e segmentos que se interagem em um ambiente físico com algumas propriedades como por exemplo a gravidade.

Para utilizar esta biblioteca é preciso criar as formas dos objetos (Shapes), associá-los a um corpo (Body) e adicioná-los ao ambiente físico (Space). TODO: Explicar melhor cada conceito com códigos.

Existe uma versão do Chipmunk portado para a linguagem Ruby.

2.4 Animação com Gosu

A animação é feita com o Gosu, uma biblioteca para desenvolvimento de jogos em 2D em Ruby e C++. Fornece somente as seguintes funcionalidades:

- Criação de uma janela com um laço principal e callbacks
- Criação de textos e imagens 2D
- Som e música em vários formatos
- Tratamento de eventos de entrada de teclado e mouse

TODO: Mostrar exemplos com códigos

2.5 Cenários com Glade

Glade é uma ferramenta que facilita o desenvolvimento de interfaces de usuário em formato de XML. A interface inicial do usuário foi criado com Glade.

TODO: Falar melhor do Glade, visualruby e mostrar exemplos de código

2.6 Exemplo

Mostrar codigos da simulação.

3 Discretização da simulação

3.1 Tempo de simulação

Tempo de simulação é o tempo utilizado para avançar a simulação de física por uma pequena quantidade de tempo.

3.2 Tempo fixado

3.3 Tempo variável

4 Colisões

4.1 Algoritmos para detecção

- Geometria computacional
- Alguns algoritmos: BSPTree, QuadTree
- Comparação

4.2 Colisões no Chipmunk

4.3 Efeito de colisões

- ? - Ajuste de colisão vs Correção dos objetos colididos.
- * A posteriori (discrete) versus a priori (continuous) collision detection

5 Atividades realizadas

Nesta seção explicamos o processo da realização do trabalho, desde o desenho do projeto no papel e ideias iniciais até a criação dos ambientes físicos de demonstração, interface de criação de cenários (Physimulation) e finalmente as integrações com exercícios-programas.

5.1 Estudo inicial

Após nosso orientador nos apresentar as discussões e ideias do Grupo de Apoio ao BCC (seção 1 e 11), decidimos planejar nosso trabalho de formatura levando em conta esta preocupação com os alunos do BCC. Inicialmente, nosso foco seria nas disciplinas de física (FAP-126) e estatística (MAE-121).

Partimos para a leitura de artigos e páginas da internet que tivessem relação com esta ideia. A seguir listamos algumas das referências mais interessantes neste processo:

- *An Introduction to Computer Simulation Methods: Applications to Physical Systems*, Gould, Tobochnik
- *Evaluation of real-time physics simulation systems*, Boeing e Bräunl
- *Esquema de detecção e resposta a colisões para animação física simplificada*, Temistocles, Atencio
- Box2D, (TODO site e explicação)
- Hotruby, (TODO site e explicação)

Percebemos que muitos estudos já havia sido feito na área de simulações e animações físicas, motivados principalmente pela criação de jogos para computador, *video-games* e celulares. Neste último campo, com a recente expansão do uso de *smartphones*, havia um número considerável de bibliotecas voltadas para reprodução de ambientes físicos. Durante esta fase pesquisa, o projeto começava a tomar mais forma e descartamos a possibilidade de trabalhar com a disciplina de estatística. A partir deste momento, concentraríamos esforços apenas em física.

Pesquisamos então as bibliotecas *open-source* disponíveis e adequadas ao nosso objetivo de integração com a disciplina do IME. Como já havíamos decidido trabalhar com a linguagem Ruby (seção 2), procuramos por *frameworks* nesta linguagem. Nesta fase de pesquisa, as bibliotecas que selecionamos para nosso trabalho foram o Chipmunk e Gosu (seção 2).

Além dos *frameworks* vistos na seção anterior, outras bibliotecas foram consideradas para o projeto. Por exemplo...

Porém...

Então...

Figura 2: Listhings - Aplicativo do Google Chrome para organização de tarefas

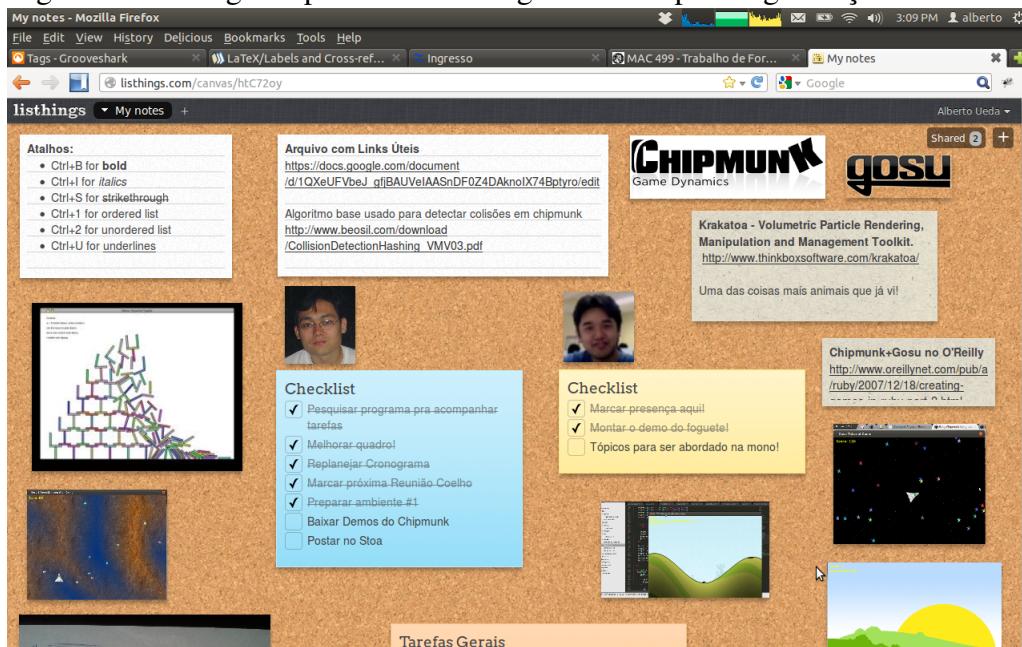


Figura 3: Sublime Text 2 - Editor de Texto feito em Python

The screenshot shows the Sublime Text 2 interface with the following details:

- File Bar:** ~/lme/click/simulation/bin/scenario_creator.rb (tcc) - Sublime Text 2 (UNREGISTERED)
- Folders:** A sidebar listing project structure:
 - tcc
 - Monografia
 - Simulation
 - bin
 - demos
 - config
 - images
 - media
 - Cannonball.rb
 - LunarLanding.rb
 - RocketSimulation.rb
 - Rockets.rb
 - ValleyBall.rb
 - lib
 - physics.rb
 - tests
 - config
 - config_com_lag.rb
 - config_exemplo.rb
 - config_gerado.rb
 - config_gravitacao.rb
 - config_gravitacao_2.rb
 - config_pir.rb
 - config_two_balls_two_.rb
 - media
 - Test.rb
 - .vr_settings.yaml
 - README.md
 - main.rb
 - chipmunk
- Code Editor:** ScenarioCreator.gladex scenario_creator.rb x
 - Code content (partial):

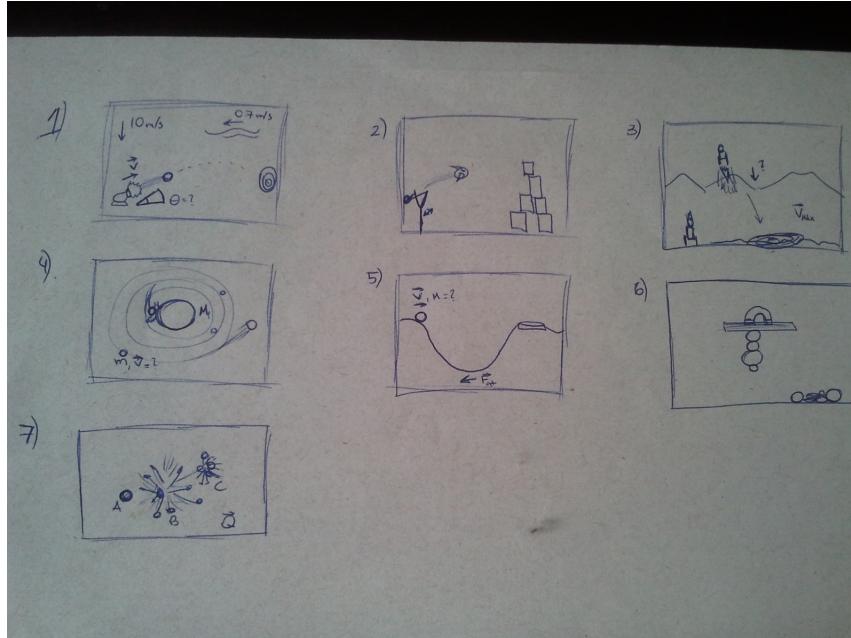
```
1 require 'chipmunk'
2 require 'gosu'
3
4 class ScenarioCreator
5   include GladeGUI
6
7   def show
8     load_glade(_FILE_)
9     set_glade_all
10    show_window
11  end
12
13  def demobutton_clicked(*args)
14    @selected_config_file = @builder['input_file_button'].filename
15
16    if (@selected_config_file)
17      use_config_file
18    else
19      read_space_data
20      read_rectangles_data
21      read_triangles_data
22      read_segments_data
23      generate_config_file
24    end
25
26    run_simulation
27  end
28
29  def read_space_data
30    @gravity_x = @builder['gravity_x'].text.to_i
31    @gravity_y = @builder['gravity_y'].text.to_i
32    @damping = @builder['damping'].text.to_f
33    @limited_space = value:@builder['limited space']
34  end

```
- Status Bar:** line 10, Column 20 | Tab Size: 4 | Ruby

5.2 Ambientes de demonstração

Após definida a linguagem e as ferramentas que utilizariamos no trabalho, partimos...

Figura 4: Primeiros rascunhos das demonstrações

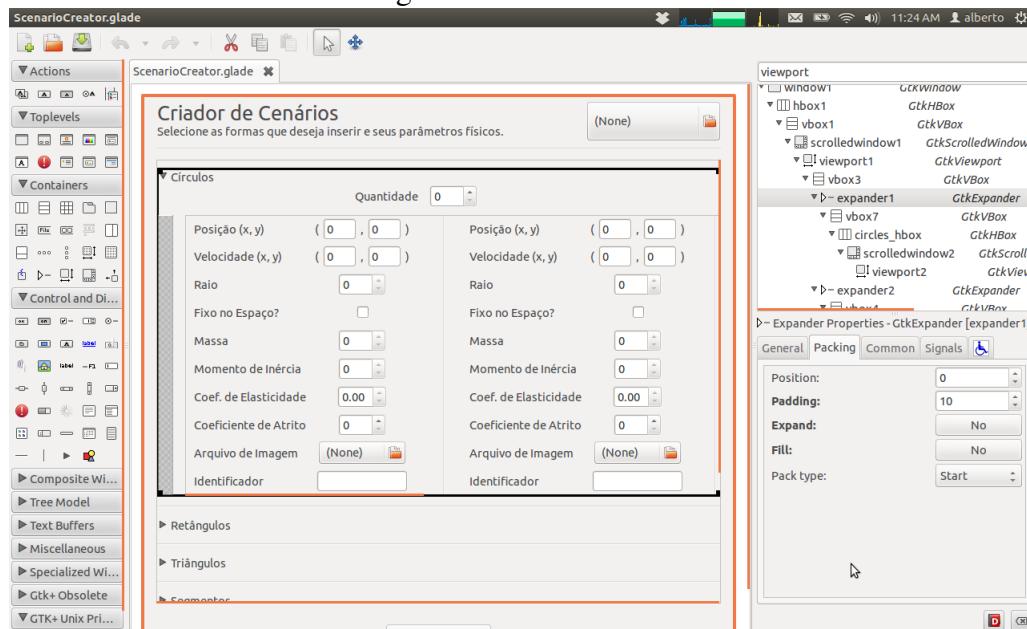


5.3 Criador de Cenários

Com um conhecimento mais profundo das bibliotecas após a implementação dos ambientes de demonstração, começamos a montar a interface de criação de cenários físicos. Como já tínhamos esta ideia em mente desde o início do projeto, centralizamos boa parte do código que era importante em classes que facilitariam a criação de simulações genéricas. Esta estratégia nos facilitou bastante o trabalho desta etapa.

Encontramos o glade...

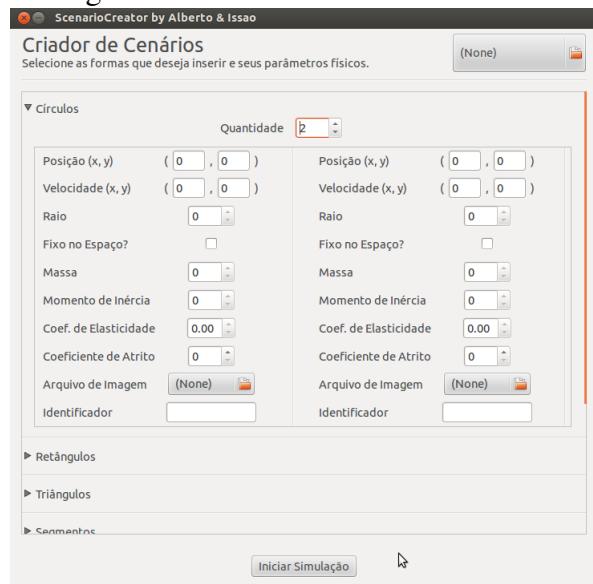
Figura 5: Gnome Glade



Discutímos durante as reuniões com o orientador e o João Kerr como seria a interface, as propriedades físicas que poderiam ou não poderiam ser alteradas...

Falar da questão do momento de inércia...

Figura 6: Interface ao final das discussões



6 Animações produzidas

As animações de diferentes tipos de ambientes físicos foram nossos primeiros resultados obtidos. Após desenhar alguns cenários simples que ilustravam tópicos da disciplina de física, como descrito na seção 5, partimos para implementação. Nossa principal objetivo nesta etapa era compreender melhor o funcionamento do Chipmunk: seus conceitos, seus recursos e suas limitações.

Além disso, como tínhamos em mente implementar um criador de cenários, a medida que implementávamos estas demonstrações extraímos todo código relevante para classes auxiliares (TODO nome melhor?) que pudessem ser reutilizadas posteriormente. Atualmente, a classe `physics.rb` contém grande parte deste código extraído.

colocar trecho de código `physics.rb`

6.1 Descrição



Figura 7: Cannonball

Cannonball foi nosso primeiro cenário físico. Baseado em jogos comuns de tiro ao alvo, como arco-e-flecha e catapultas, o objetivo é acertar uma bola de canhão em um alvo de posição escolhida aleatoriamente. O usuário pode modificar a angulação do canhão e atirar a bola pressionando a tecla de espaço. Há um vento de intensidade também calculada de forma aleatória, na direção horizontal e sentido contrário ao do trajeto da bola ao alvo. As informações relevantes ao usuário - ângulo do canhão, intensidade do vento, pontuação, entre outros - são exibidas no topo da janela.

Nossa intenção com este cenário era ilustrar o comportamento de lançamentos oblíquos e resistência do ar.

Posteriormente, utilizamos o código de *Cannonball* na integração com exercício-programa *Angry-Bixos*, descrita na seção 8.

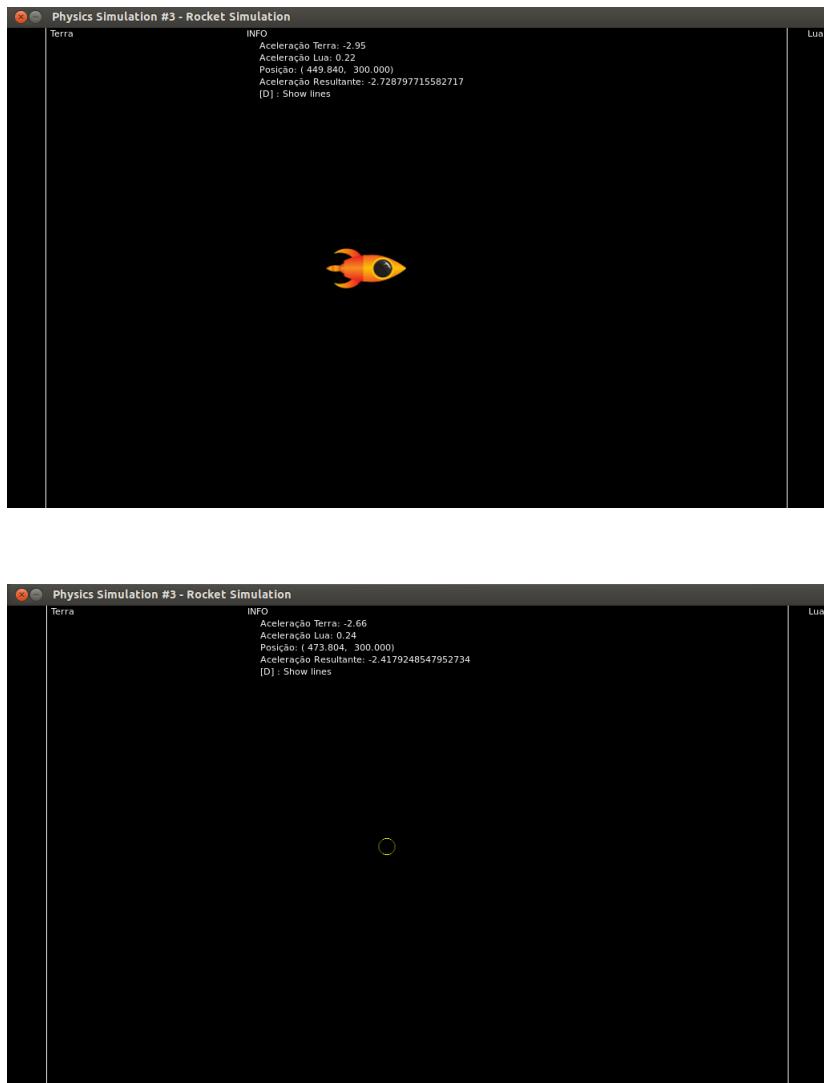


Figura 8: Rocket Simulation

A ação conjunta das gravidades da Terra e da Lua sobre um objeto é simulada em *Rocket Simulation*. Controlando um foguete, o usuário pode movimentar-se com as setas do teclado entre os dois corpos, representados por retas nas extremidades esquerda e direita da tela. Por ter uma massa bem maior, a atração exercida pela Terra é maior em boa parte da tela, exceto em regiões bem próximas a Lua.

Por motivos de simulação, não reproduzimos os valores reais de massa de ambos os corpos, pois a região em que a Lua teria maior efeito sobre a nave seria ínfima e não proporcionaria ao usuário a experiência que gostaríamos.



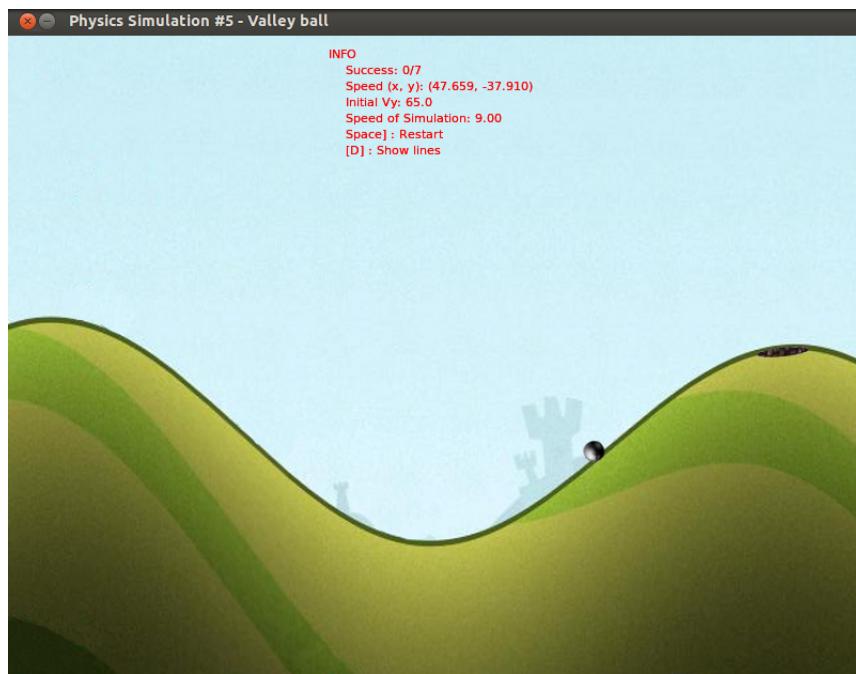
Figura 9: Lunar Landing

Baseado no jogo *Lander* (TODO url na bib), o objetivo é aterrissar um foguete

em uma superfície plana com uma velocidade bem pequena. Se o usuário não conseguir diminuir adequadamente a velocidade antes de tocar a superfície, o foguete explode. Em caso de sucesso, ele pousa e uma mensagem de sucesso é exibida. As setas do teclado movimentam o foguete.

Com a implementação desta demonstração utilizamos em conjunto o código de detecção de colisão do *Cannonball* com a manipulação de objetos de *Rocket Simulation*.

Tanto neste e em outros cenários há a função "Raio-X" (tecla 'd'), que mostra ao usuário o esqueleto dos objetos do cenário. Em outras palavras, os *shapes* que estão ativos no *space* da simulação. Mais detalhes na seção 2.



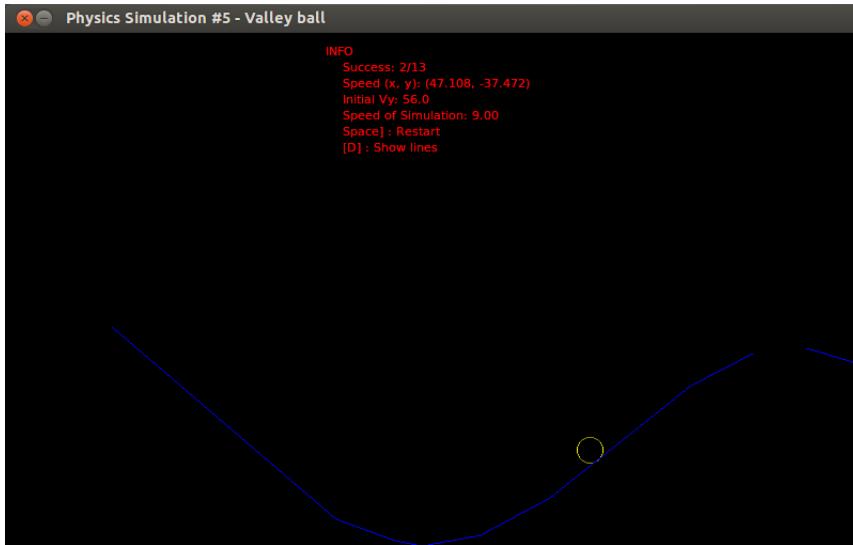


Figura 10: Valleyball

Em *Valleyball*, tentamos ilustrar o Princípio de Conservação da Energia Mecânica e a aplicação da força de atrito. Uma bola é lançada a uma certa altura do chão com velocidade vertical determinada aleatoriamente. Ela então desliza sobre um pequeno vale e dependendo da velocidade inicial ela pode: 1) voltar ao vale e perder velocidade até a situação de repouso; 2) cair em um buraco localizado próximo ao topo de uma montanha; ou 3) prosseguir seu movimento para fora do cenário. O que determina a situação resultante é a velocidade inicial da bola. A tecla de espaço reinicia a simulação com uma nova velocidade para a bola.

Nesta demonstração utilizamos pela primeira vez o controle de velocidade da simulação. Dentro de alguns limites, o usuário pode alterar o tamanho do passo da simulação (seção 3), tornando a animação mais rápida ou mais lenta. Isso é útil, por exemplo, para acelerar alguma etapa da simulação cujos resultados sejam previsíveis ou pouco interessantes.

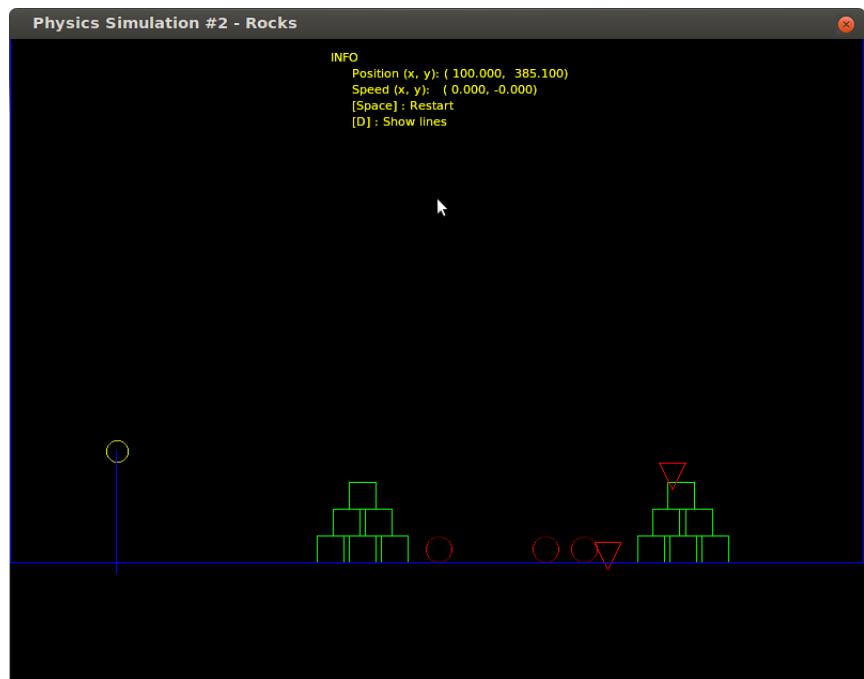


Figura 11: Rocks

Rocks foi nosso último cenário de demonstração. Parecido com *Cannonball*,

o objetivo também é acertar alvos da tela - neste caso os objetos de cor vermelha. Porém é o usuário que determina a força que a bola será atirada, dependendo de quanto ele ”puxar” a bola segurando e arrastando o *mouse*, de modo semelhante ao comportamento de um estilingue. Com este cenário físico gostaríamos de simular um jogo que recentemente ficou bastante conhecido, o *Angry Birds*. Além dos alvos, há cubos empilhados na tela, sendo estes de dois tipos: de pequena massa (cor de madeira) e de grande massa (cor de pedra). O resultado das colisões entre os objetos da tela variam de acordo com a massa do objeto.

Além de ser o único cenário de demonstração que possui interação com o *mouse*, *Rocks* consolidou nosso conhecimento das bibliotecas utilizadas e das modificações que precisávamos fazer para implementar o Physimulation. Em termos de física, a novidade deste cenário é a utilização de Energia Potencial Elástica no lançamento do objeto.

6.2 Visualização das animações

Para visualizar as animações descritas nesta seção, o aluno deve executar o seguinte comando:

```
$ cd <<DIR_PHYSIMULATION>>/Simulation
$ ruby main.rb demos
```

onde *DIR_PHYSIMULATION* é o diretório em que o Physimulation foi instalado. Em seguida, deve escolher uma das animações listadas no programa e clicar em ”Iniciar”. Caso haja algum problema ao executar qualquer um dos comandos, verificar o conteúdo da seção 10, ”Instruções de instalação da plataforma”.

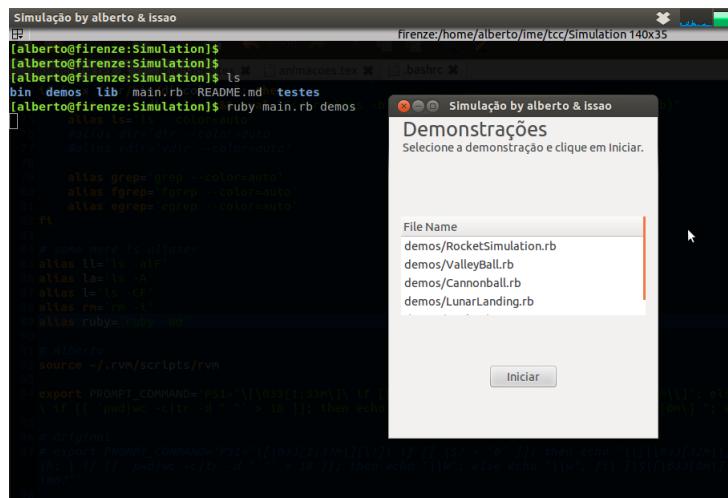


Figura 12: Menu de demonstrações

7 Physimulation

Descrição da interface de criação de cenários e alguns exemplos do que pode ser criado com ele.

7.1 A ferramenta

7.2 Criando um cenário básico

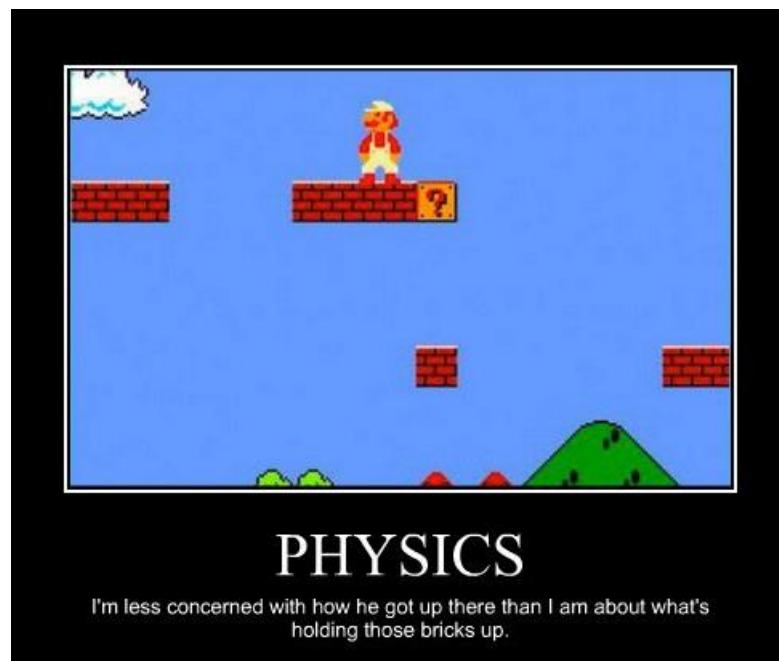


Figura 13: Objetos fixos

7.3 Exemplos

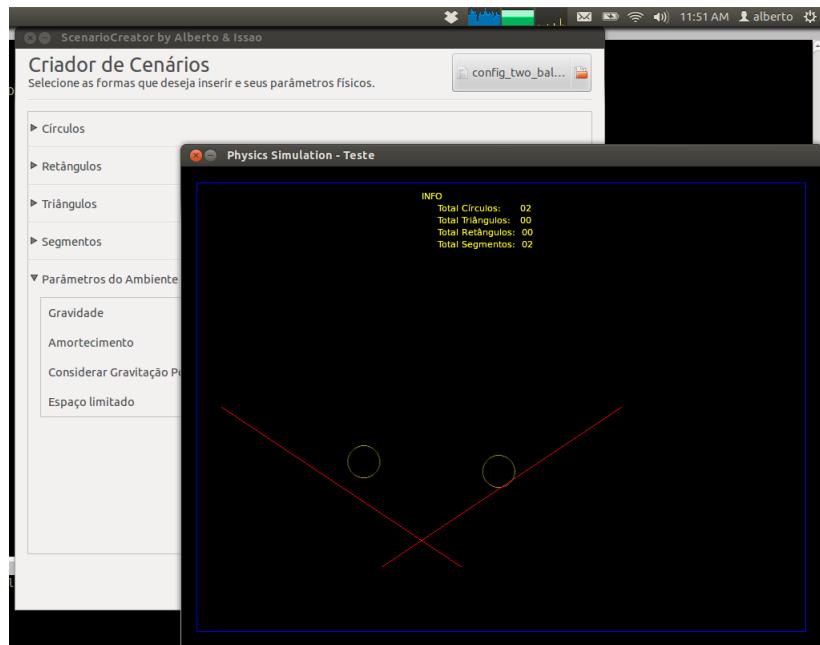


Figura 14: Cenário 1

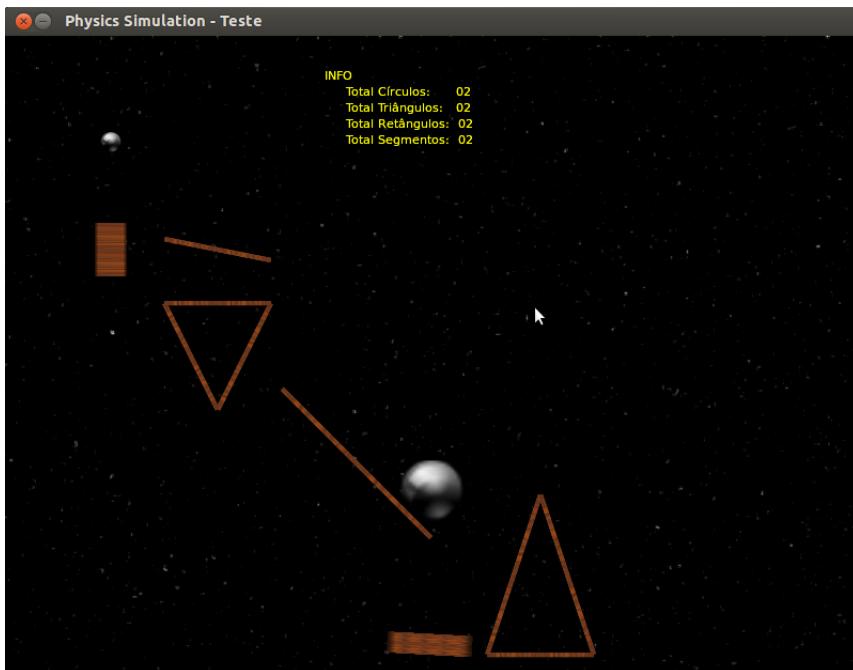


Figura 15: Cenário 2

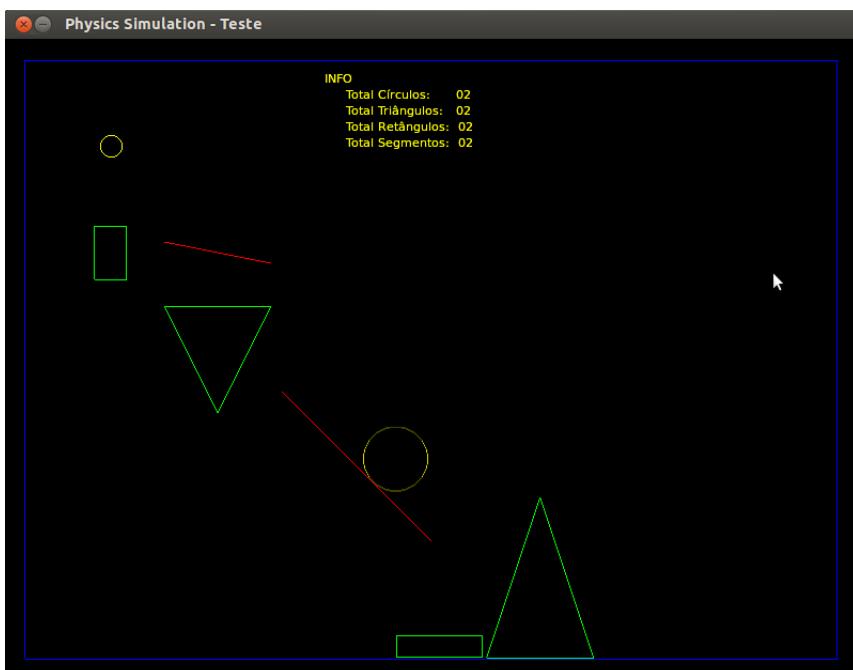


Figura 16: Cenário 2 - Raio-X

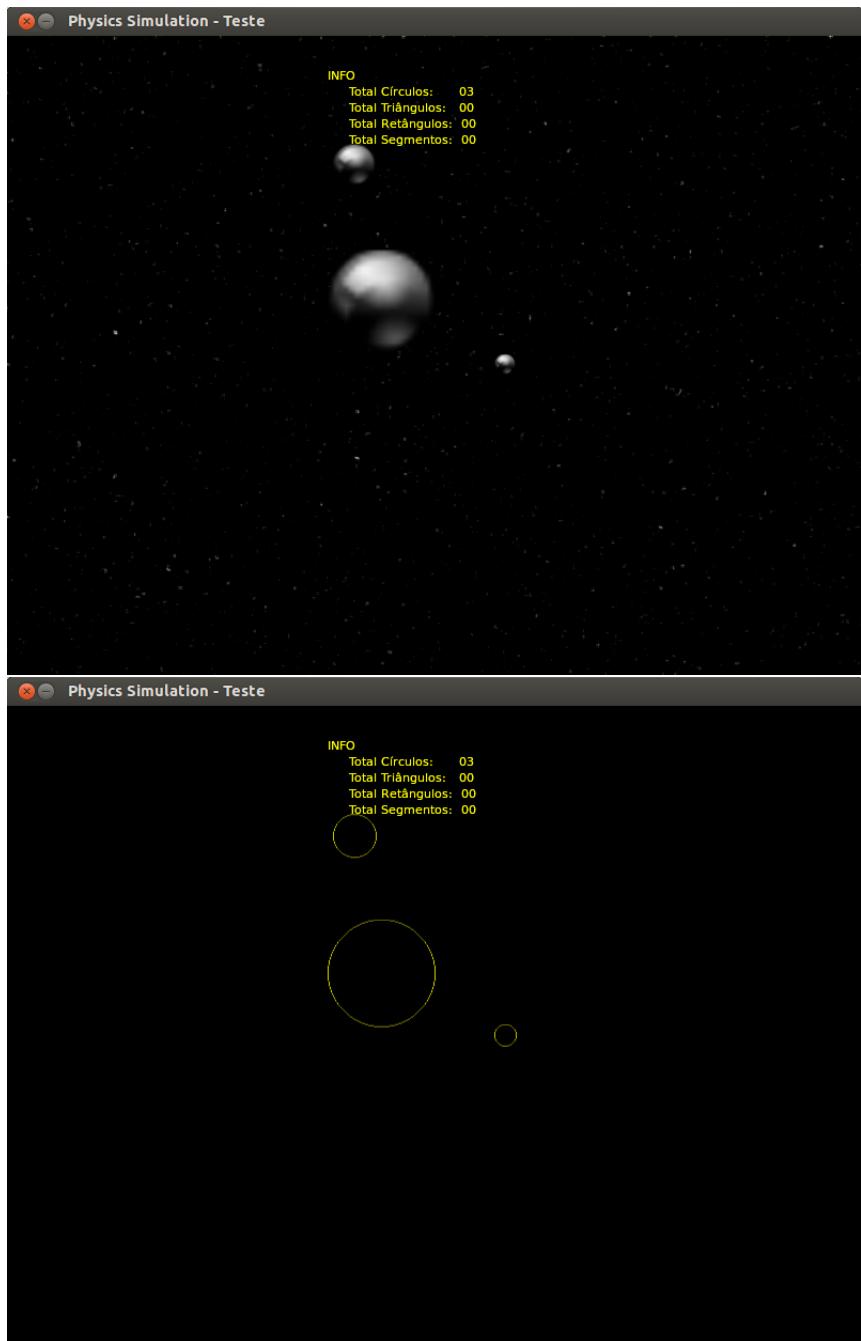


Figura 17: Cenário Gravitação 1

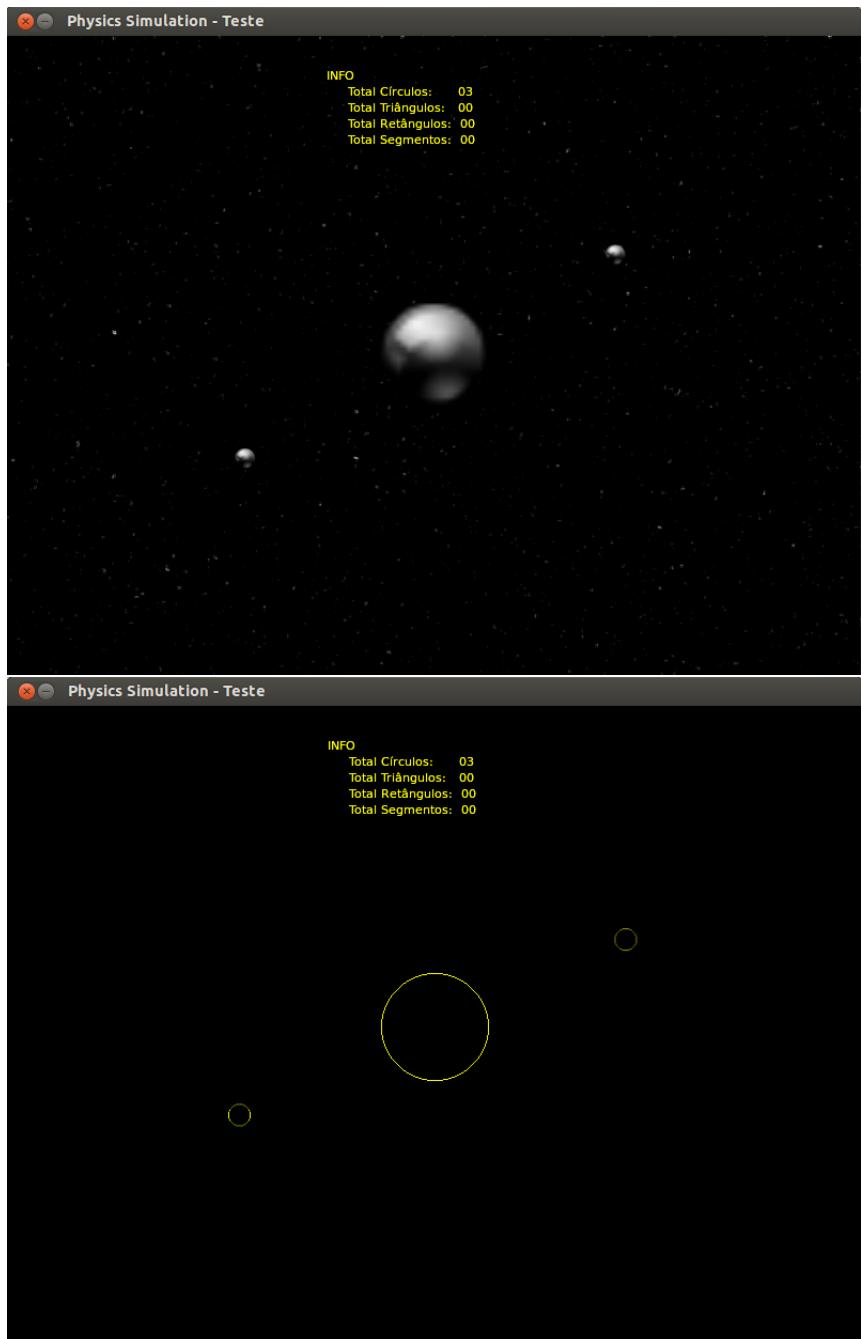


Figura 18: Cenário Gravitação 2

7.4 Arquivos de configuração

As propriedades físicas do cenário e de cada objeto ficam guardadas em arquivos *.config e podem ser modificadas...

7.5 Pontos de modificação

Apesar de utilizar o Chipmunk, o Physimulation possui pontos do código que podem ser modificados de acordo com o comportamento físico que deseja-se obter, o que pode ser interessante para o aluno de computação.

Tais pontos são...

Além disso, no próprio Chipmunk há trechos de código que o aluno poderá modificar e observar rapidamente mudanças na simulação. Exemplos: ...

8 Introdução à computação com animações

Nesta seção explicamos as duas integrações que fizemos com exercícios-programas dados em disciplinas de introdução a computação e que envolviam física.

8.1 Configuração

8.2 Angry Bixos

8.3 Apolo

9 Comentários finais

9.1 Trabalhos futuros

9.2 Desafios

9.3 Conclusão

10 Instruções de instalação da plataforma

Passos para instalação das bibliotecas: ruby, chipmunk, gosu, chingu, etc.

Falar do warning do próprio chipmunk: gems/ruby-1.9.3-p286/gems/chipmunk-5.3.4.5/lib/chipmunk.rb:6: Use RbConfig instead of obsolete and deprecated Config

(Talvez seja um apêndice)

11 Apêndice

O texto a seguir foi extraído do site: <http://bcc.ime.usp.br>.

Material de Apoio ao BCC

A falta de contextualização das disciplinas básicas do BCC tem sido uma queixa recorrente dos alunos nas reuniões entre alunos e professores, no Encontro do BCC de 2010 e também no processo de avaliação semestral que é realizado pelo orientador pedagógico orientador pedagógico da Escola Politécnica (POLI), Giuliano Salcas Olguin.

A fim de motivar os alunos e ilustrar a relação entre ciência da computação e as disciplinas básicas de álgebra, cálculo, estatística, probabilidade e física presentes no currículo do BCC a CoC sugeriu que fossem produzidos documentos ilustrando aplicação de cada uma dessas disciplinas em ciência da computação e vice-versa. Esses documentos têm o objetivo de motivar os alunos do BCC:

1. ilustrando as relações entre as disciplinas básicas do curso e ciência da computação;
2. mostrando aos alunos quais das disciplinas mais avançadas do BCC que fazem uso dos conteúdos das disciplinas básicas;
3. fornecendo aos professores das disciplinas básicas do BCC exemplos de aplicações de suas especialidades em ciência da computação, que, eventualmente, podem ser mencionados em aulas ou ser temas de trabalhos.

Esses documentos poderão também ser usados pelas disciplinas de Introdução à Ciência da Computação que são oferecidas pelo DCC para várias unidades da USP. Nestas disciplinas, frequentemente, os chamados exercícios programas ilustram aplicações de métodos computacionais na solução de problemas em genética, física, economia, etc. Por exemplo, na última edição da disciplina

MAC2166 Introdução à Ciência da Computação para Engenharia

podemos ver um exercício programa em que é simulada a "trajetória de livre de retorno" de uma nave sob a ação gravitacional da Terra e da Lua em <http://www.ime.usp.br/~mac2166/ep3/>. Já um exercício programa com aplicação em genética pode ser visto em <http://www.ime.usp.br/~mac2166/ep4/>.

Além de uma maior integração do curso este projeto pretende propor possíveis mudanças na grade curricular do BCC. Para isto pretendemos realizar uma pesquisa com o egresso do BCC e uma pesquisa das grades curriculares dos cursos de computação pelo mundo.

Parte II

Parte subjetiva

Nesta seção descreveremos a relação entre nosso projeto e a experiência adquirida no BCC.

Entregar este projeto como trabalho de formatura e disponibilizar seu código para os alunos do BCC foram duas das experiências mais gratificantes que já tivemos. Isto pois acreditamos que tal conteúdo poderá ser utilizado pelas próximas turmas do BCC como incentivo ao aprendizado da matéria de física. Além disso, tanto alunos do próprio Instituto de Física quanto da Engenharia Politécnica também poderão se interessar pelo conteúdo: o primeiro grupo (FIS) pela animação de fenômenos físicos estudados e o segundo (Poli) tanto pela animação quanto pela simulação de tais fenômenos.

Mas, ao mesmo tempo, por ser um trabalho que levou meses, certas dificuldades foram encontradas pelo caminho. Tivemos que tomar decisões às vezes frustrantes, porém necessárias.

11.1 Desafios e frustrações encontrados

Inicialmente, nossa motivação era entregar um sistema que utilize recursos do Wii Remote (TODO ref TODO link da caneta) e que o professor pudesse utilizá-lo em sala de aula para realizar suas simulações e animações. Porém, chegamos a conclusão que esta tecnologia aumentaria consideravelmente o nível de complexidade de nosso trabalho e não tínhamos garantia de que utilizá-la acrescentaria da mesma forma ao resultado final. Assim descartamos esta possibilidade.

Como utilizamos algumas bibliotecas de terceiros em nosso projeto, tivemos que entender obrigatoriamente como eram feitas as principais chamadas de métodos destas bibliotecas, principalmente o Chipmunk e o Gosu. Um detalhe interessante que ocorreu no segundo mês de trabalho foi a necessidade de mudar o código da biblioteca (TOOD citar) e recompilá-la para que uma função simples de mensagem para o usuário funcionasse (TODO conferir método). Uma semana depois, utilizando uma versão mais nova da biblioteca, descobrimos que nossa alteração não era mais necessária, pois já havia sido feita pelos próprios programadores na mudança de versão.

Além disso, utilizamos um binding (TODO envoltório?) da versão original do Chipmunk. Isto trazia duas dificuldades para nós: 1) o código original (em C++) sempre estava com uma versão mais recente e provia (TODO conferir) mais métodos; e 2) nem sempre o que víamos na documentação oficial possuia correspondente em nosso binding.

Por último, um desafio que tivemos foi encontrar um professor de física disponível para nos auxiliar na elaboração do protótipo do sistema. Ficamos muito felizes quando após algumas semanas o bacharel em física e aluno do BCC João Kerr veio a uma de nossas reuniões, a convite do professor Coelho.

11.2 Disciplinas mais relevantes - Alberto Ueda

- MAC0110 Introdução à Computação : Embora já tivesse contato com programação no ensino técnico, foi nesta disciplina que passei a conhecer e utilizar boas práticas de programação. Além disso, estudei algoritmos famosos e interessantes (por exemplo os de ordenação) que estimularam-me para as disciplinas que viriam a seguir.
- FAP0126 Física I : É a grande motivação deste trabalho. Os conceitos aprendidos nesta disciplina estão por todo nosso código e nas simulações produzidas. Com o Physimulation, tentamos unir o que vimos nesta disciplina com a computação.
- MAC0122 Princípios de Desenvolvimento de Algoritmos : O maior contato com algoritmos, dos mais simples e elegantes aos mais complexos, foi fundamental para minha formação. Primeiro porque me desafiou em certos momentos - e consequentemente me deu coragem para analisar ou implementar futuros algoritmos - e em segundo lugar pois deu-me a confiança de que gostaria de seguir carreira em computação.
- MAC0211 Laboratório de Programação I : Esta disciplina foi interessante por dois motivos: pelo estímulo ao trabalho em equipe e por nos apresentar conceitos e ferramentas relacionadas a qualidade de software, como o Doxygen para documentação de código. Foi nesta disciplina que apendi o que era um Makefile!
- MAC0323 Estruturas de Dados : Essencial para minha formação como cientista da computação. As estruturas aprendidas nesta disciplina - como listas ligadas e árvores - são muito comuns na programação, mesmo no mercado. Possuem vantagens e desvantagens entre si e o conhecimento de

suas propriedades assim como os algoritmos adequados para manipulá-las foram muito importantes para mim.

- MAC0420 Introdução a Computação Gráfica : Outro forte motivador para nosso trabalho. Nesta disciplina tivemos como exercício-programa a simulação de um jogo de bilhar em três dimensões. Foi uma das experiências mais gratificantes do meu BCC, pois minha dupla e eu aplicamos física em um código simples em C com algumas bibliotecas gráficas e de repente tínhamos uma simulação razoável do que ocorre na vida real. Foi quando percebi que com poucos conceitos de física podíamos reproduzir muitos fenômenos naturais, como colisões e dissipação de energia. Percebi também o quanto estes resultados me motivavam a estudar mais, tanto física quanto computação.
- FAP0137 Física II : Os tópicos desta disciplina não foram o foco deste projeto, mas foram grandes motivadores para nosso trabalho. Assim como em Física I, houve pouca contextualização do que foi estudado com o curso de computação. No futuro, temas como relatividade restrita poderão se tornar bem mais simples de se entender por meio de animações criadas pelo próprio usuário, utilizando nosso simulador.
- MAC0332 Engenharia de Software : na área de computação, um dos conceitos mais recorrentes em qualquer projeto de longo prazo é ciclo de vida de um *software*. Este era o tópico mais discutido na disciplina, tornando-a fundamental para o aluno de computação. Outro aspecto a destacar é a prática de trabalho em equipe.
- MAC0338 Análise de Algoritmos: difícil descrever em poucas linhas o quanto esta disciplina é importante para o aluno de computação. Além do desempenho ser uma preocupação constante e necessária a qualquer programador, o aprendizado nesta disciplina é uma das minhas bases sólidas como desenvolvedor. É uma das matérias que quero aprofundar meus conhecimentos durante o mestrado.
- MAC0446 Princípios de Interação Homem-computador : uma das disciplinas mais legais para o aluno que está preocupado com a usabilidade de seu sistema. Os conceitos aprendidos estão por todo o trabalho, assim como em outros projetos que participei ou fui responsável.

11.3 Disciplinas mais relevantes - Rafael Miyagawa

11.4 Estudos futuros

Sem dúvida os tópicos de estudo mais importantes para a continuação deste trabalho são as disciplinas de Física I e II para o BCC. Quanto maior o conhecimento das leis e forças físicas presentes no mundo real, melhor serão as simulações e consequentemente as animações geradas.

Em segundo lugar, seria interessante uma análise de qual das alternativas a seguir tem uma melhor relação custo-benefício, visando a atualização do projeto com a versão mais nova do Chipmunk: A) migrar nosso projeto de Ruby para C++ e usar diretamente a versão original do Chipmunk, sem bindings; ou B) atualizar o binding em Ruby adicionando os métodos e funcionalidades da versão mais recente em C++.

Por último, mas não menos importante, um estudo de paradigmas que proporcionem mais usabilidade ao usuário, substituindo o preenchimento obrigatório de formulários para criação de objetos físicos. Ex: *drag-and-drop* do mouse para "arrastar" as formas geométricas, fornecendo os valores de massa, coeficientes de elasticidade e atrito *a posteriori* (após o objeto já estar na tela).