Programação e Desenvolvimento de Software I

Pedro O. S. Vaz de Melo
olmo@dcc.ufmg.br

UFMG - ICEx
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Objetivos

- Introduzir o aluno aos conceitos de algoritmos e estruturas de dados

Objetivos e metas

- Introduzir o aluno aos conceitos de algoritmos e estruturas de dados
  - Noções da organização e funcionamento de um computador
  - Noções de linguagens imperativas
  - Noções de estruturas de dados

Por que aprender programação?

- Porque quase todas as profissões atualmente interagem com um computador

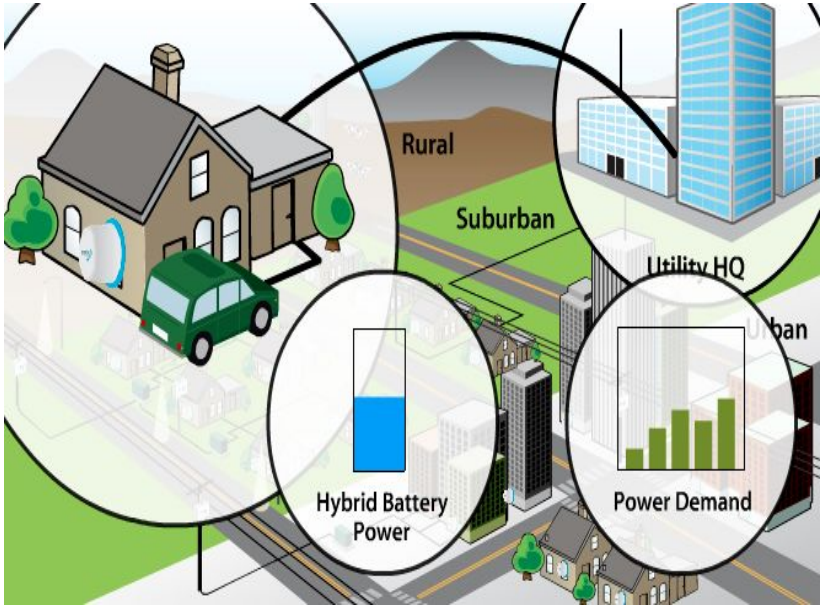
- Interação cada vez mais sofisticada
  - Administração: planilhas Excel com macros
  - Física, química, matemática: uso de ferramentas como Octave, Scilab, Matlab, …
  - Biologia: simuladores para criar novas moléculas

Por que aprender programação?

Engenharia de controle a automação: PLCs (Programmable Logic Controllers), cruise control

Por que aprender programação?

- Engenharia elétrica: smart grids, telecomunicações



Por que aprender programação?

- Será necessário para outras disciplinas
  - PDS II
  - Circuitos digitais (pode ser)
  - Análise numérica
  - Redes de computadores
  - Informática industrial
  - Controle digital
  - Pesquisa operacional
  - Optativas: bancos de dados, redes de computadores, visão computacional, …

O nosso curso

- Aulas teóricas e práticas
- Aulas teóricas:
  - Apresentação dos conceitos
- Aulas práticas:
  - Resolução de listas de exercícios
  - Programas a serem desenvolvidos ou exercícios relacionados

O nosso curso

- Aulas síncronas e assíncronas
- Aulas assíncronas:
  - Videoaulas de apresentação dos conceitos
- Aulas síncronas:
  - Resolução de exercícios e dúvidas

Aulas práticas

- Aulas no laboratório (LAICO) ou (1009 e 1010)
- Datas marcadas de azul no cronograma
- Aulas práticas com **avaliação**:
  - Entrega, até o final da aula, da solução de um problema simples de programação
  - Valor de cada entrega: 1 ponto

Aulas assíncronas

- Todos as videoaulas estão disponíveis na página da disciplina
  - vídeos no Youtube ou para download
- Acompanhe as semanas em que as videoaulas deverão ser vistas pelo:
  - cronograma
  - Moodle: 2021_2 - PROGRAMAÇÃO E DESENVOLVIMENTO DE SOFTWARE I - TF_TW - METATURMA

Aulas síncronas

- Datas marcadas de **laranja** no cronograma
- Resolução de exercícios de fixação
- Apoio a qualquer tipo de dúvida e problemas relacionados à disciplina

Avaliações

- 3 provas individuais
  - Primeira: 15 pontos
  - Segunda: 27 pontos
  - Terceira: 30 pontos
- Exercícios práticos
  - 5 pontos de práticas semanais
    - Pode ser em dupla, **mas cada um entrega o seu**
  - 3 pontos de exercícios práticos avaliativos
    - Individual
- Trabalho prático
  - 20 pontos
  - Individual
  - Possibilidade de vários pontos extras

Provas

- Conteúdo base: livro, transparências e exercícios
- Dou dicas sobre a prova durante as aulas
  - Comentários em sala de aula
  - Exercícios parecidos
- A resposta pode não estar no material, mas a matéria lecionada é a base teórica para resolvê-la

Provas

- Assíncronas
- Duração: 48 horas
- Serão feitas no VPL do Moodle
  - Ambiente virtual de programação com correção automática
  - Com consulta, mas individual

Provas

- Síncronas (presenciais)
- Duração: 1 hora e 40 minutos
- No papel no VPL do Moodle (a definir)
  - Ambiente virtual de programação com correção automática

Revisão da correção das provas

- Até duas semanas depois da entrega da nota da avaliação
  - Evitar choradeira no fim do semestre!
  - Se gostou da nota quando recebeu a correção, tem que gostar dela no fim do semestre também!
  - Importante revisar, pois a correção automática pode levar a erros injustos (ex: na sexta casa decimal)

Práticas regulares

- Entrega no Moodle: arquivos .c e .h
- Pontuação do tipo “entregou, levou”
- Há vídeos com as correções de todos os exercícios práticos

Práticas avaliativas

- Avaliativas (treino para a prova)
- Correção automática no VPL do Moodle
- Pontuação depende da correção das questões
- Mesmo formato das provas

Trabalho prático

- Momento para realizar um projeto mais longo e complexo
- Julgamento do código
  - Comentários, facilidade de leitura, indentação
- Julgamento da documentação
  - Estruturação, clareza e coesão,conteúdo

Algumas ideias sobre o trabalho

"É uma excelente oportunidade para desenvolver a capacidade de aprender novas ferramentas."

Algumas ideias sobre o trabalho

"Permite que o aluno se divirta desenvolvendo um trabalho prático."

Algumas ideias sobre o trabalho

"Permite o desenvolvimento de um trabalho maior, mais complexo, além de incentivar a criatividade do aluno."

Algumas ideias sobre o trabalho

"TP totalmente não condizente com a realidade, fruto de um devaneio de um profissional frustrado."

Avaliação do trabalho prático

- Nota final =
  (
  até 20 pontos para o trabalho básico considerando os itens mencionados anteriormente
  + pontos extras
) * nota da prova oral

Avaliação do trabalho prático

- Prova oral consistirá de X (ex: 3) perguntas diretas sobre o código
- Nota da arguição = número de perguntas respondidas corretamente em tempo hábil dividido por X
  - Ex: se x = 3, valores possíveis: 1, 2/3, 1/3, 0
- Se você sabe o que fez no TP, vai tirar 1 na prova oral!
- No entanto, vários alunos já tiraram 0 e 1/3 na prova oral :~(

Avaliação do trabalho prático

- Pontos extras só serão dados aos alunos que ficarem com mais de 50% dos pontos **nas provas** (mais de 36 pontos)
  - 36 não é > 50%, não adianta chorar!

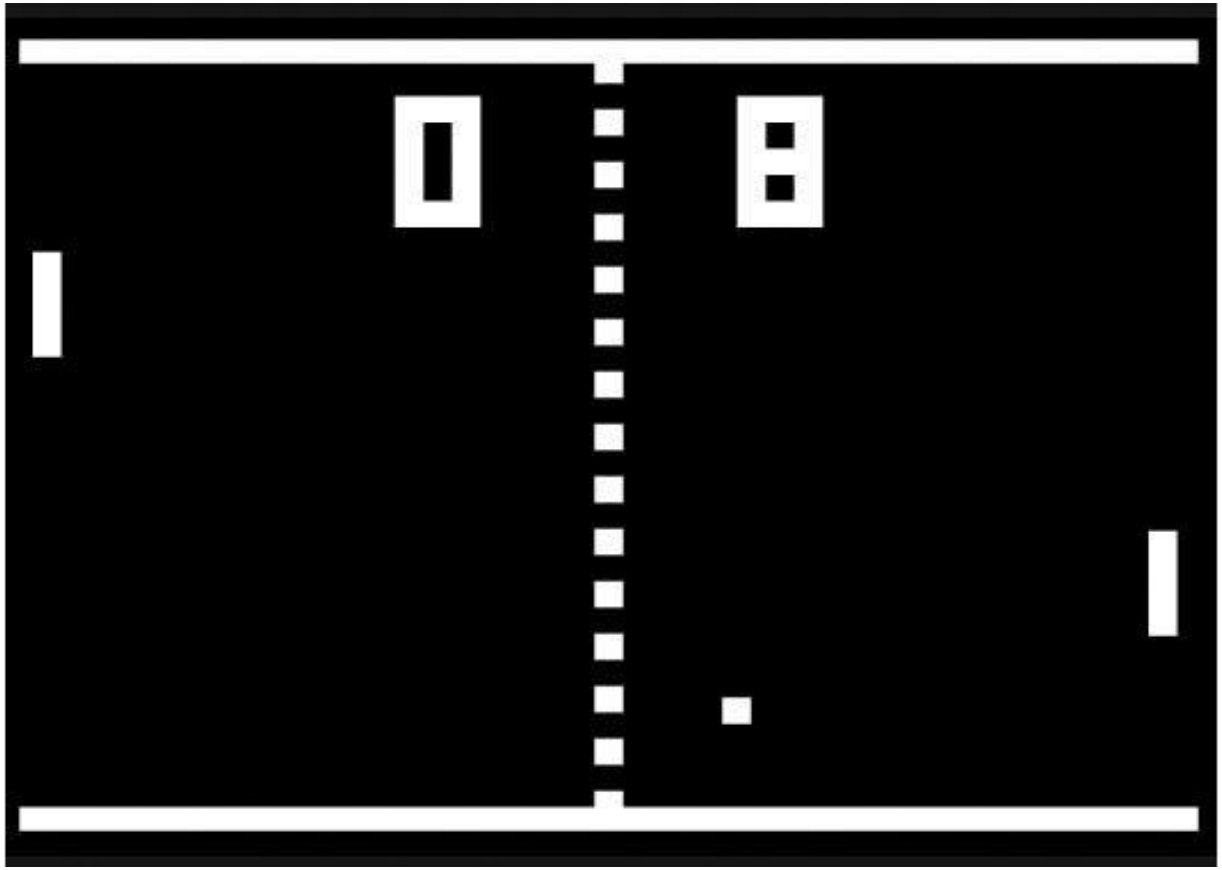
- Pontos extras SÓ servem para aumentar o conceito de D para C, C para B e B para A

- Ou seja, não espere passar com os pontos extras (E para D)!

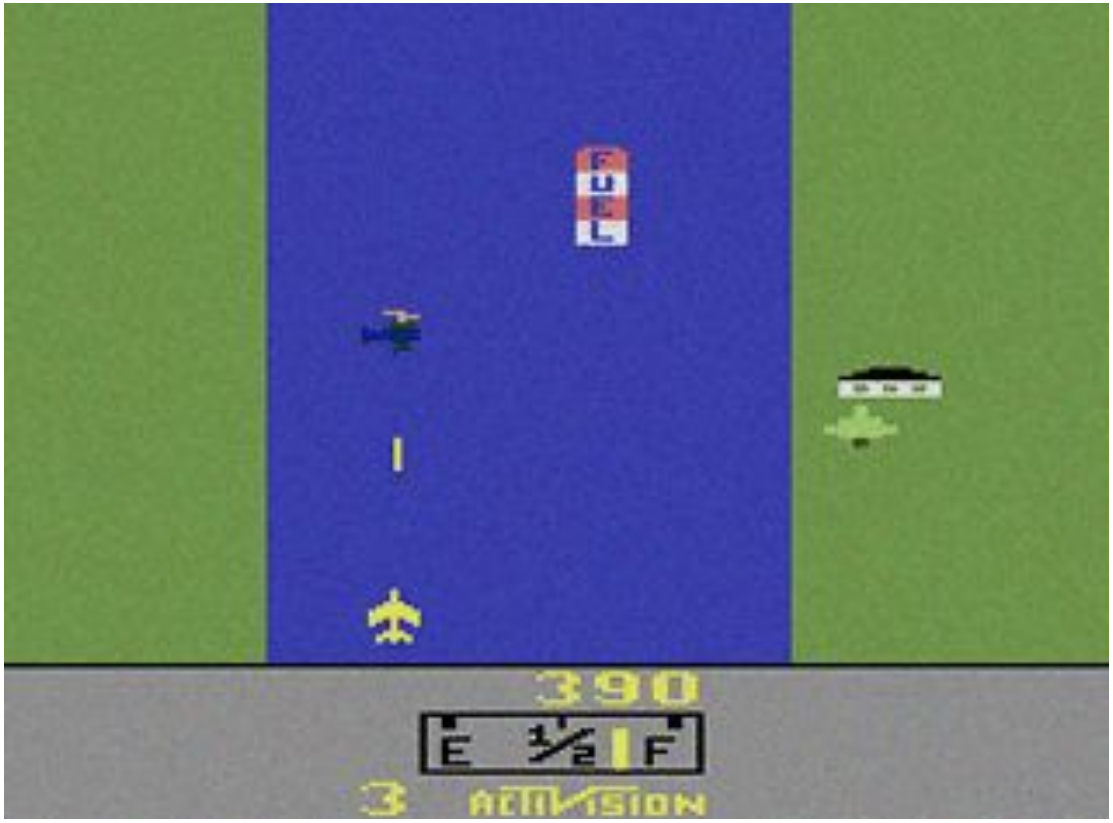
Trabalho prático do semestre 1/2013

Trabalho prático do semestre 02/2013

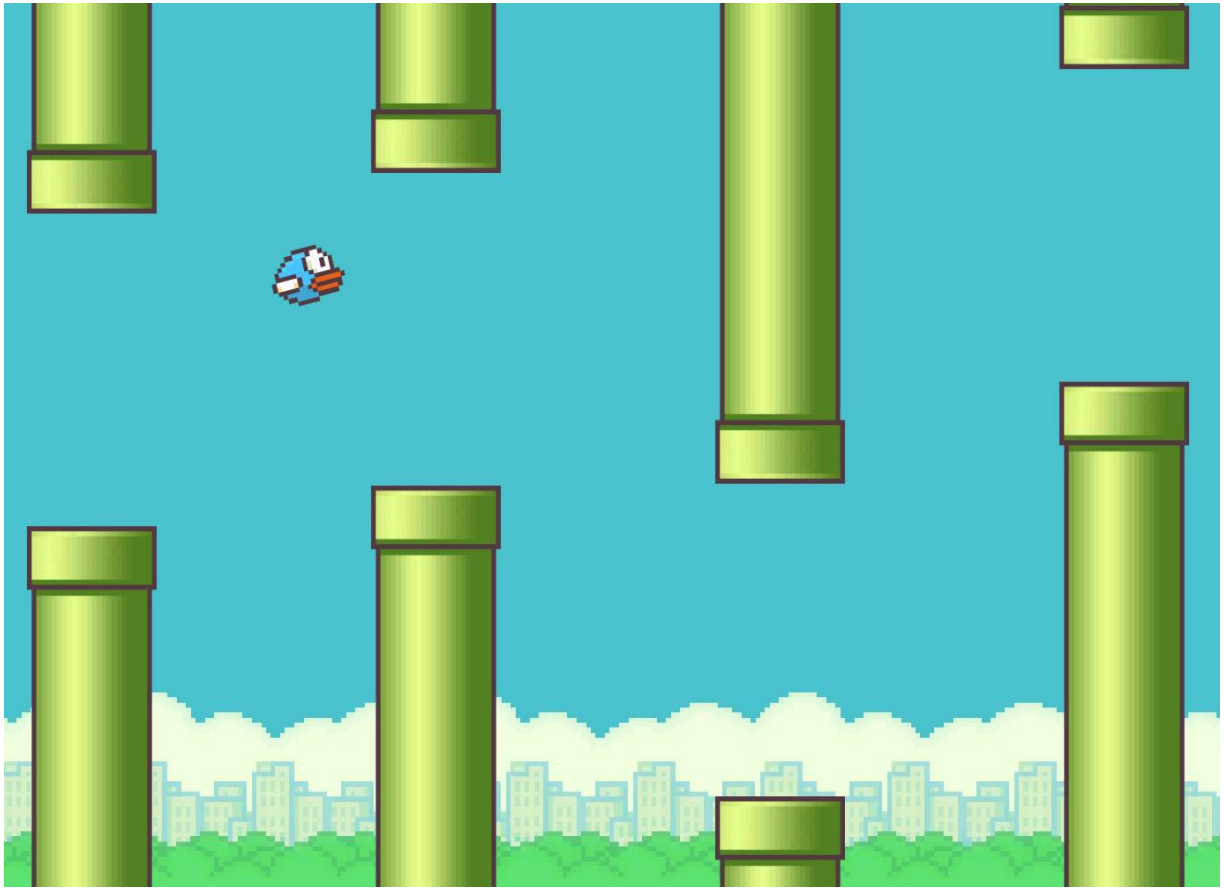
Trabalho prático do semestre 01/2014



Trabalho prático do semestre 02/2014



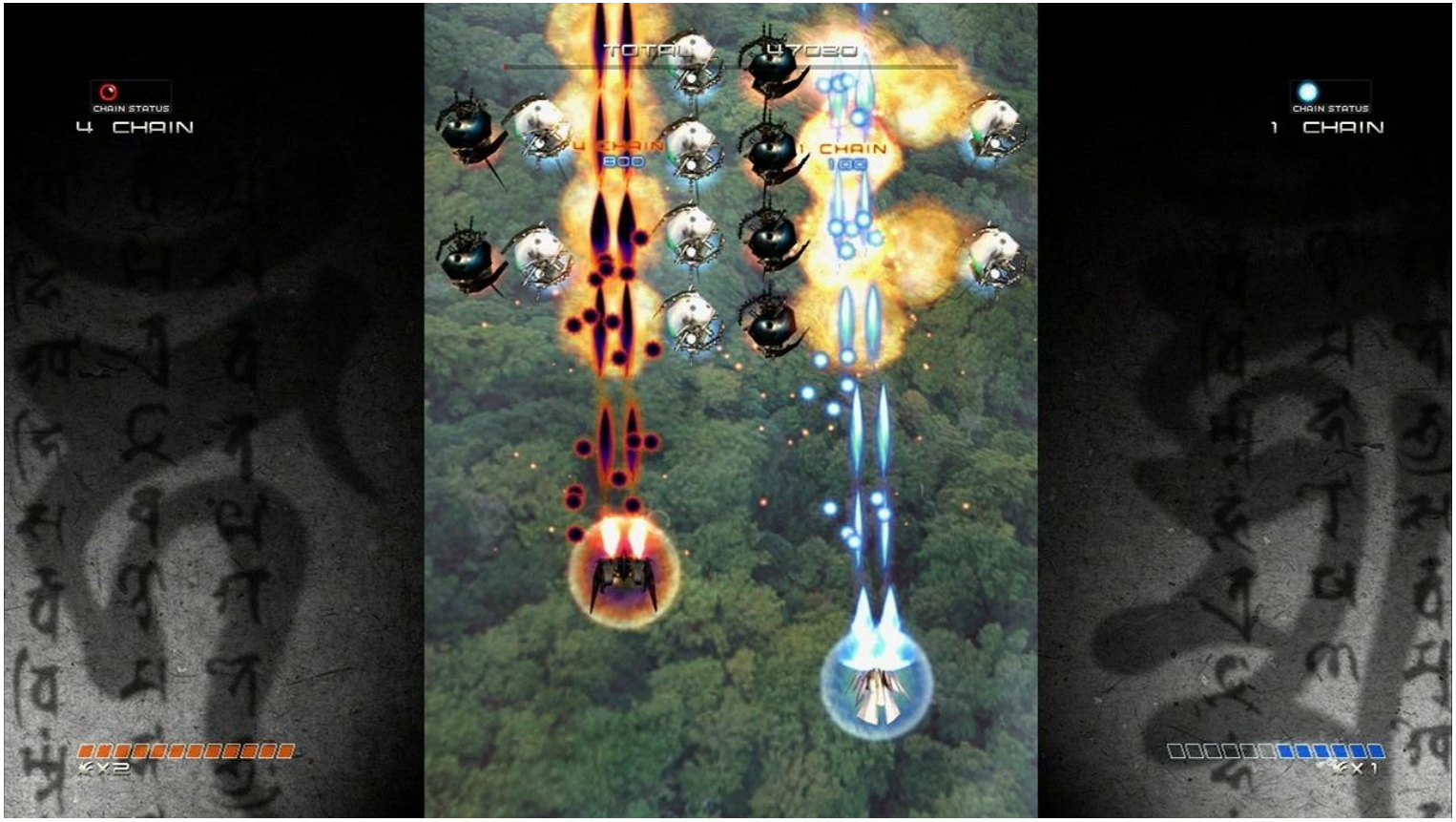
Trabalho prático do semestre 01/2015



Trabalho prático do semestre 02/2015

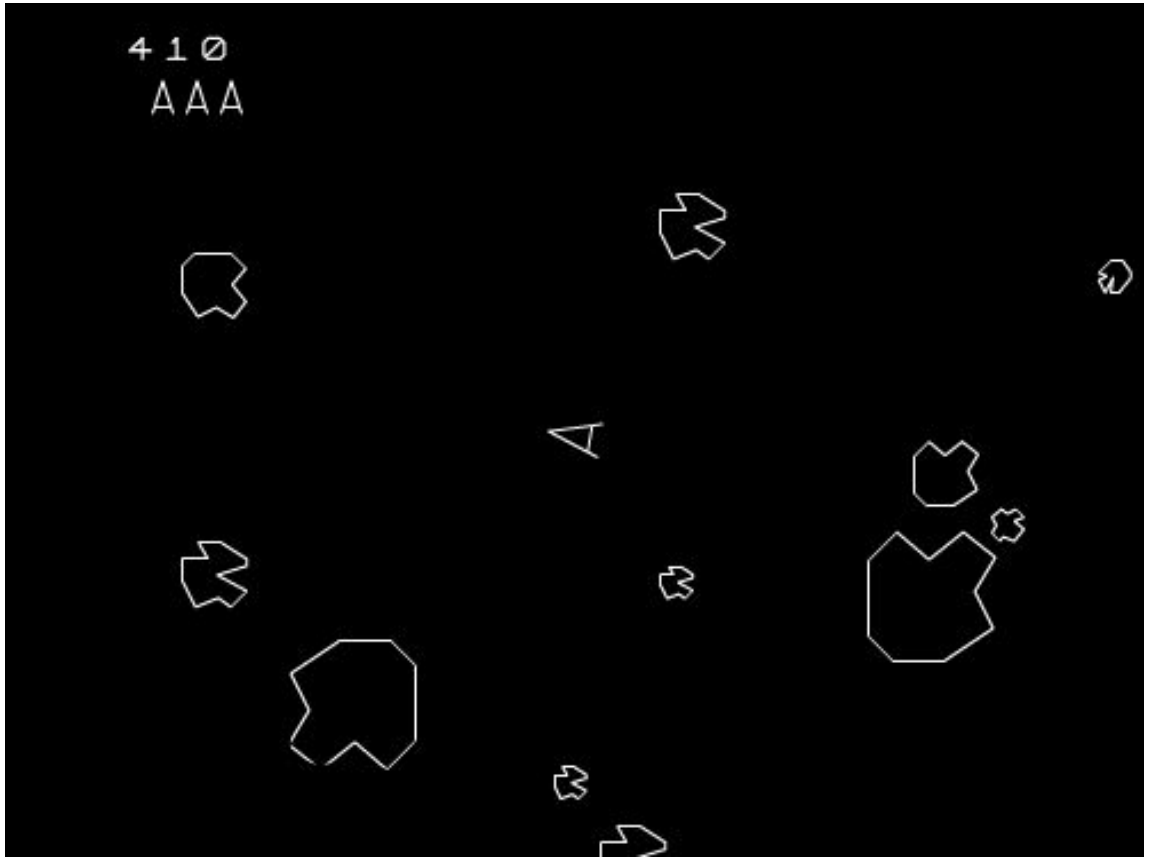
Trabalho prático do semestre 01/2016

Trabalho prático do semestre 02/2016



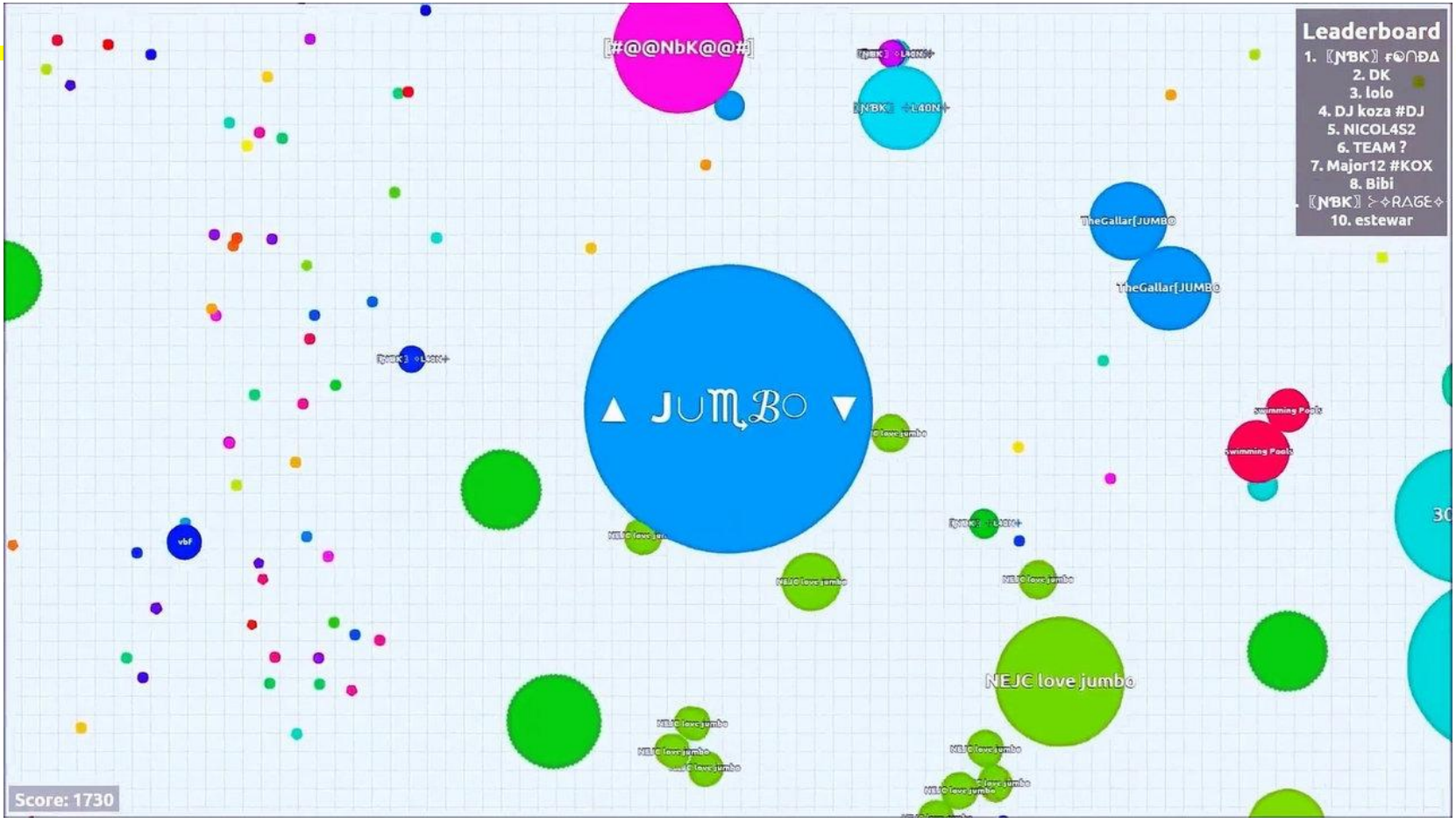
ikaruga

Trabalho prático do semestre 01/2017



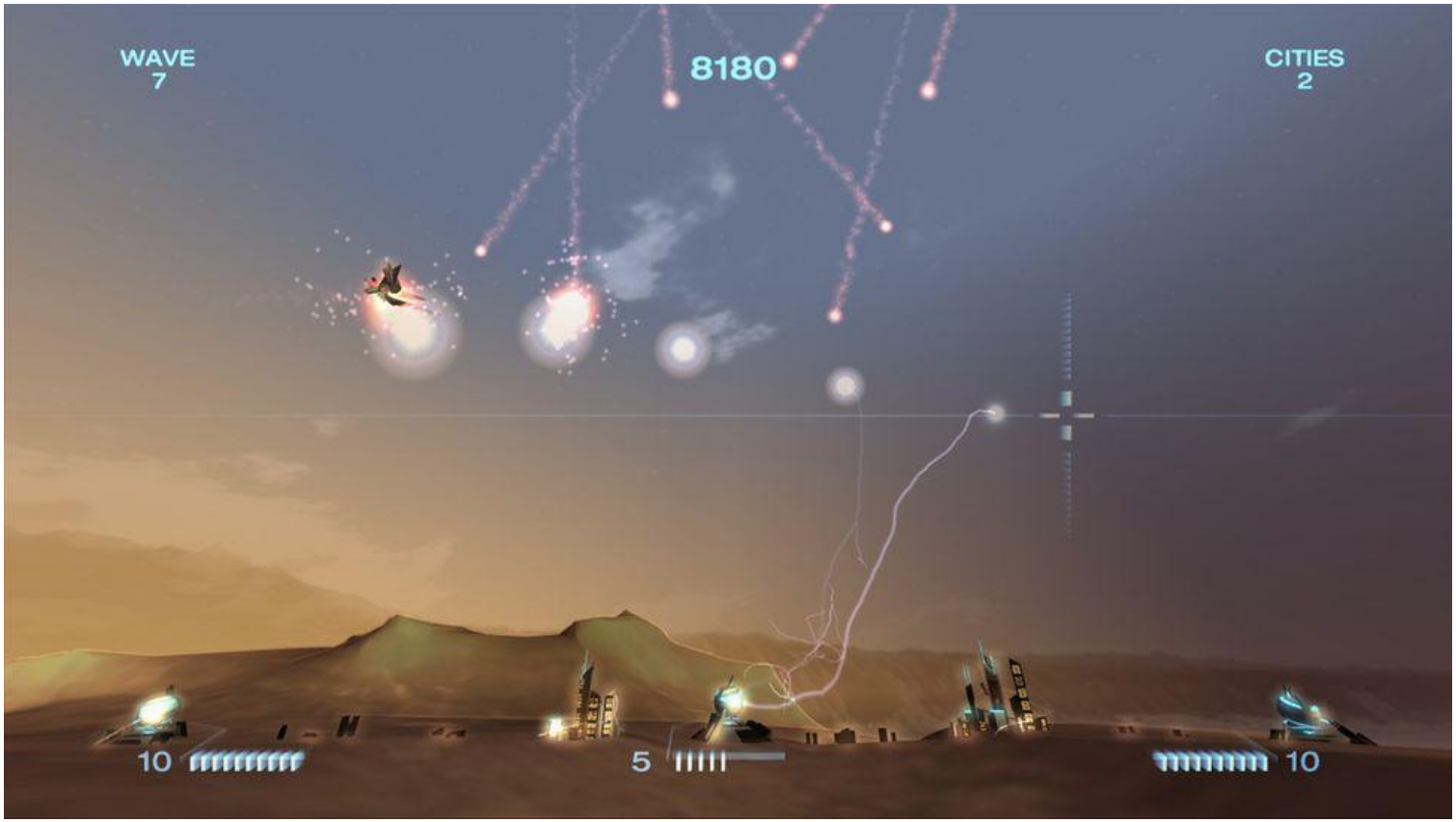
Asteroids

Trabalho prático do semestre 02/2017



Agar.io

Trabalho prático do semestre 01/2018



Missile Command

Trabalho prático do semestre 02/2018

Enduro (Atari)

Trabalho prático do semestre 01/2019

Guitar Hero

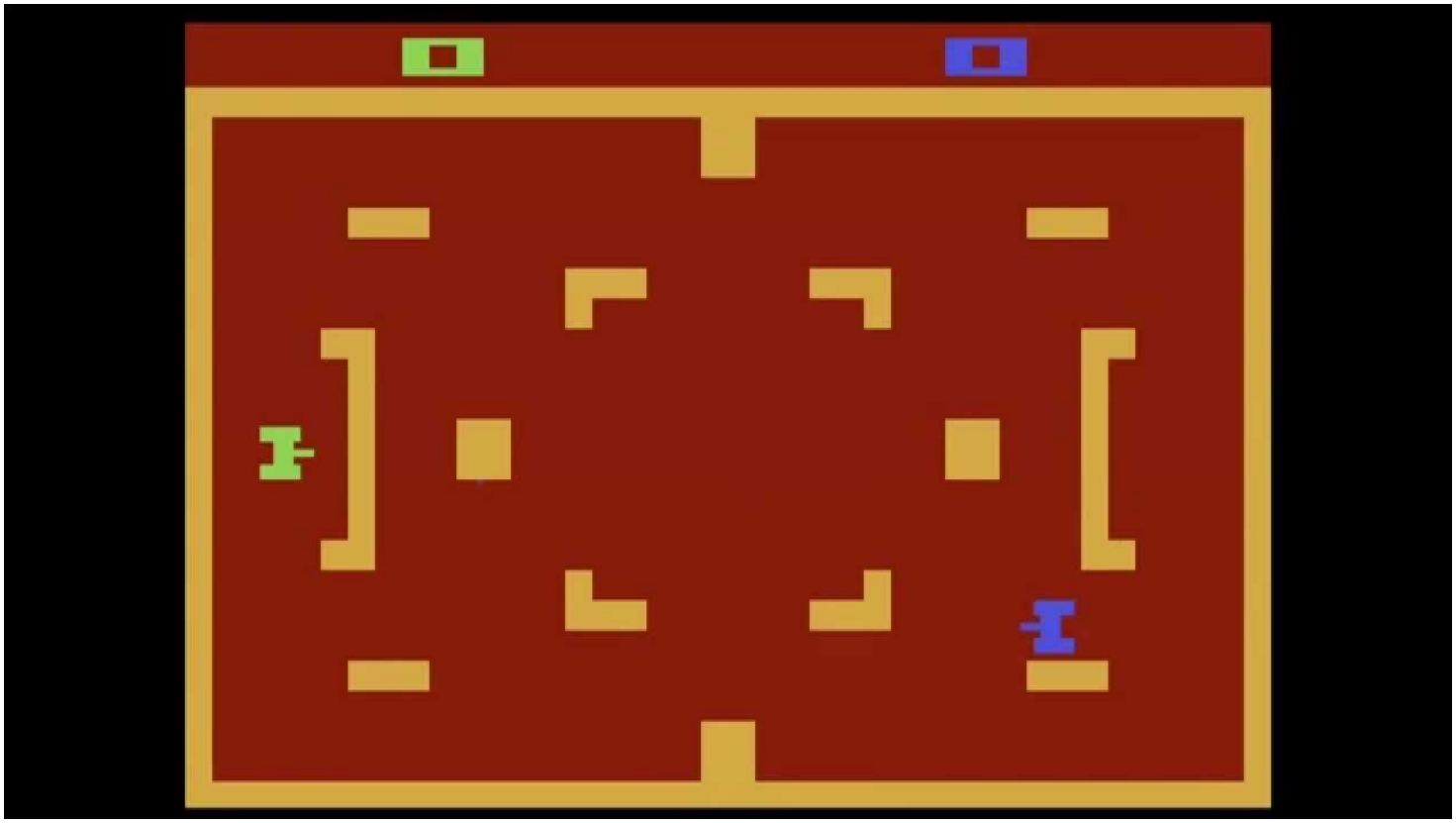
Trabalho prático do semestre 02/2019

Candy Crush

Trabalho prático do semestre 01/2020

Space Invaders

Trabalho prático do semestre 02/2020

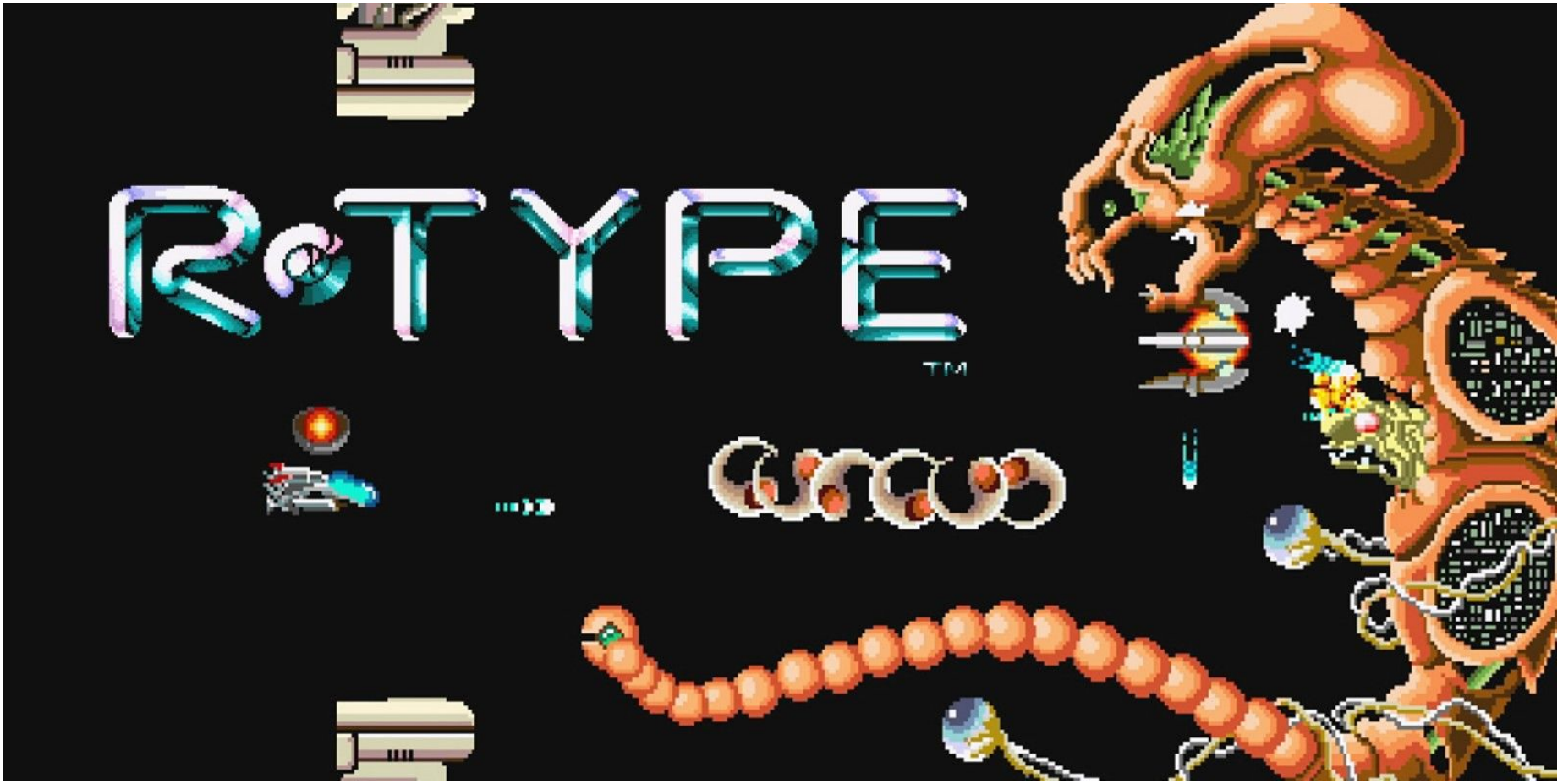


Combate

Trabalho prático do semestre 01/2021

Final Fantasy VI

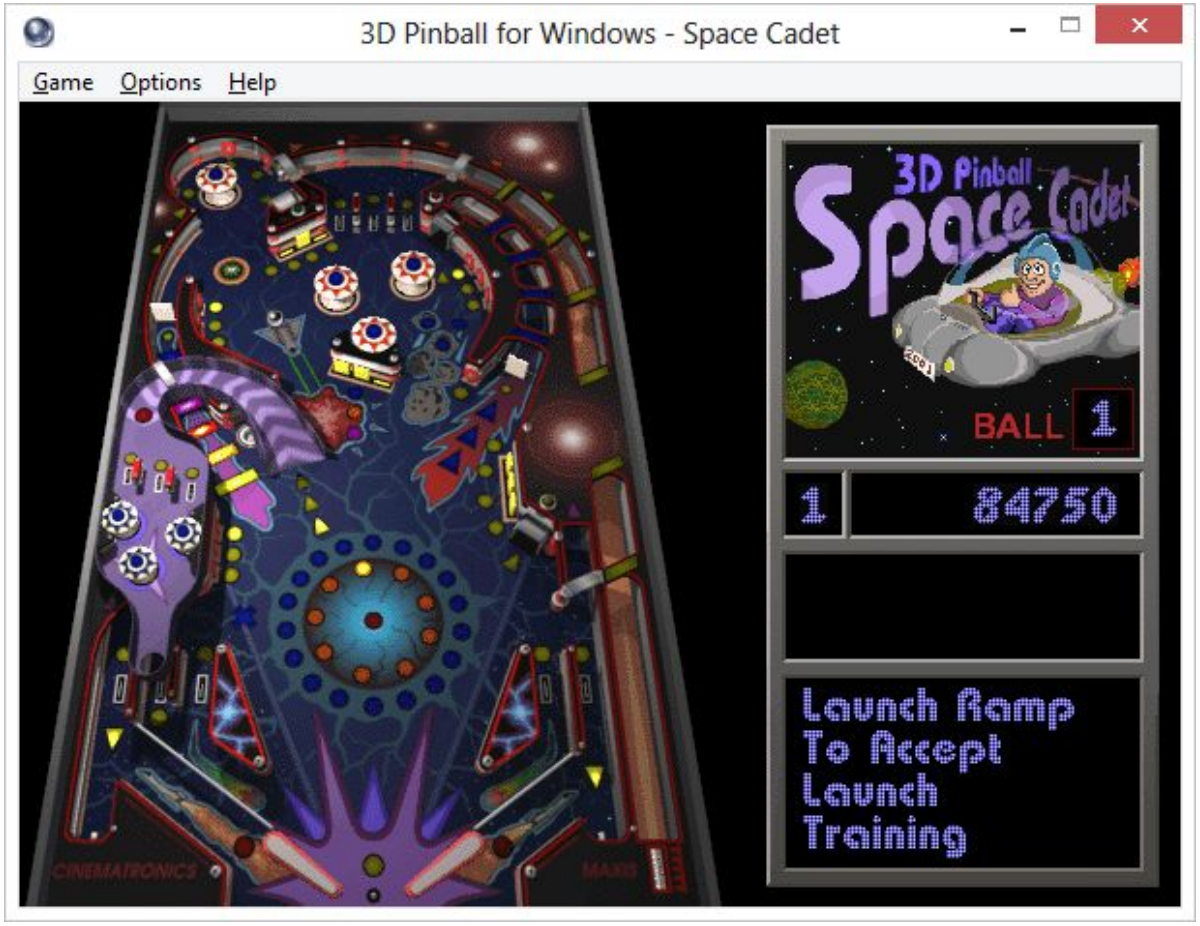
Trabalho prático do semestre 02/2021



R-Type

UFMG - ICEx
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Trabalho prático deste semestre



[3D Pinball Space Cadet (Windows)]

Motivação

tec

'Flappy Bird' chega a 50 milhões de downloads e rende US$ 50 mil por dia

DE SÃO PAULO

08/02/2014 12h10

Recomendar 178 Tweetar 61 +1 4 OUVIR O TEXTO + Mais opções

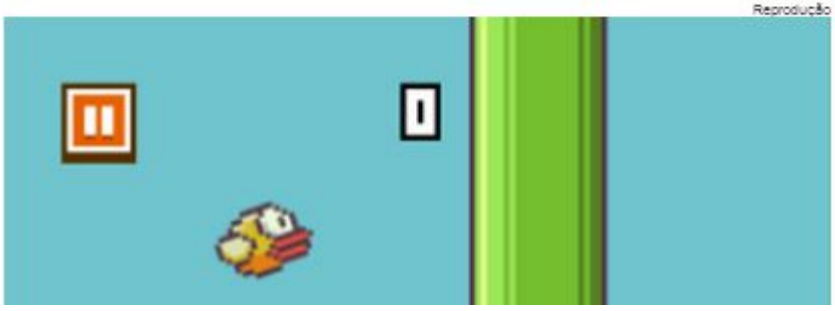
Com uma dificuldade absurda disfarçada em gráficos simples, o jogo "Flappy Bird" virou fenômeno.

Primeiro em diversas categorias na App Store e no Google Play, ele já foi baixado 50 milhões de vezes e acumula mais de 47 mil avaliações.

Dong Nguyen, o vietnamita que criou o jogo, diz faturar US$ 50 mil por dia com publicidade.

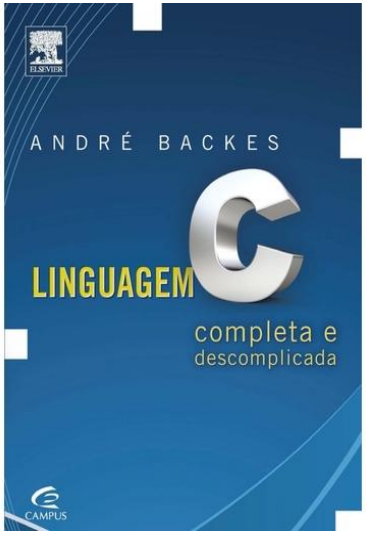
O desenvolvedor é independente e atingiu o sucesso sem a ajuda de um grande estúdio. ★★★

Reprodução



Bibliografia

- Livro-texto:
  - Linguagem C completa e descomplicada, André Backes



- Outros:
  - Introdução às Estruturas de Dados, Waldemar Celes
  - Projeto de Algoritmos com implementação em PASCAL e C, 3a edição, Nivio Ziviani
  - Algoritmos estruturados, 3a edição, Harry Farrer, Becker, Faria, Matos, dos Santos, Maia

Linguagem C

- Criada em 1972
  - 47 anos de idade
  - Para computação → bem velho
- Existem linguagens mais recentes? Sim.

Linguagem C

- Existem linguagens mais recentes? Sim
  - A escolha de uma linguagem para aprender a programar é um problema complicado
  - C serve para entender o funcionamento do computador melhor
    - Baixo nível
  - Com o tempo alunos podem usar o conhecimento para aprender outras linguagens
    - Depois de bastante prática, chavear linguagens é algo simples
  - Plano didático da UFMG é quase todo em C

Notas e frequência

- Não reprovo por frequência
  - **SE** o aluno tiver aproveitamento superior a 60%

- Se for infrequente (frequência < 75%):
  - Não ajudo a mudar de conceito
  - Não ajudo a passar (mesmo que seja por 1 ponto)
  - Não tem direito a exame especial
  - Infrequente e nota menor que 60: conceito F

Notas e frequência

- Listas de presença em todas as aulas
- Não precisa vir à aula
  - Se quiser estudar em casa
  - Se quiser ficar conversando com os colegas

É só tirar 60 pontos ou mais…

Notas e frequência

- As aulas são importantes:
  - Posso dar dicas sobre questões de prova
  - Posso propor um exercício parecido ao da prova
  - Posso mencionar algo que não está no livro, mas cai na prova

Frequência

- Não há faltas no ensino remoto

Exame especial

- Especial, como diz o nome
  - Difícil: matéria do semestre inteiro
  - Muitas questões: preciso avaliar a matéria como um todo

→ Sugestão: evitem fazer o exame especial
  → É mais fácil passar com as provas e trabalhos
  → Férias começam mais cedo

Extra-classe

- Teremos monitores
- Estou disponível fora do horário de aula
  - Marcando horário
  - Por e-mail: melhor para notas/correções
- Fórum do Moodle/Minha UFMG: dúvidas sobre a matéria/programação

É fácil aprender programação?

Teach Yourself Programming in Ten Years

Peter Norvig

Why is everyone in such a rush?

Walk into any bookstore, and you'll see how to *Teach Yourself Java in 24 Hours* alongside endless variations offering to teach C, SQL, Ruby, Algorithms, and so on in a few days or hours. The Amazon advanced search for [title: teach, yourself, hours, since: 2000 and found 512 such books. Of the top ten, nine are programming books (the other is about bookkeeping). Similar results come from replacing "teach yourself" with "learn" or "hours" with "days."

The conclusion is that either people are in a big rush to learn about programming, or that programming is somehow fabulously easier to learn than anything else. Felleisen *et al.* give a nod to this trend in their book *How to Design Programs*, when they say "Bad programming is easy. *Idiots* can learn it in *21 days*, even if they are *dummies*." The Abtruse Goose comic also had their take.

Translations

Thanks to the following authors, translations of this page are available in:

Arabic
(Mohamed A. Yahya)

العربية

Bulgarian
(Boyko Bantchev)

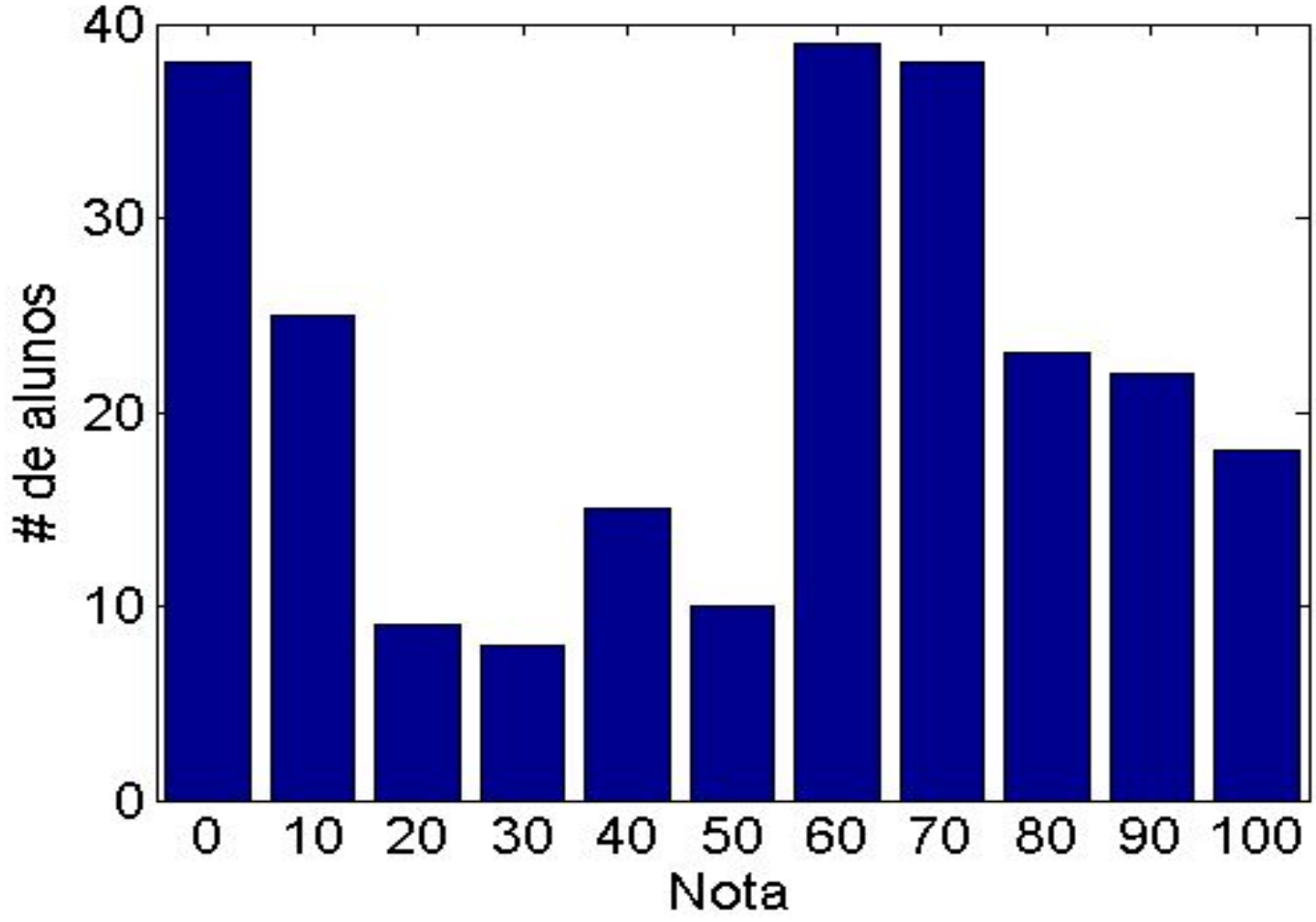
Observações

- O curso não é fácil
- Diferente do segundo grau: objetivo é formar analistas, não recitadores
- Programação é difícil de aprender
  - Diferente de tudo o que vocês já aprenderam antes
  - Matemática + lógica + “Arte”

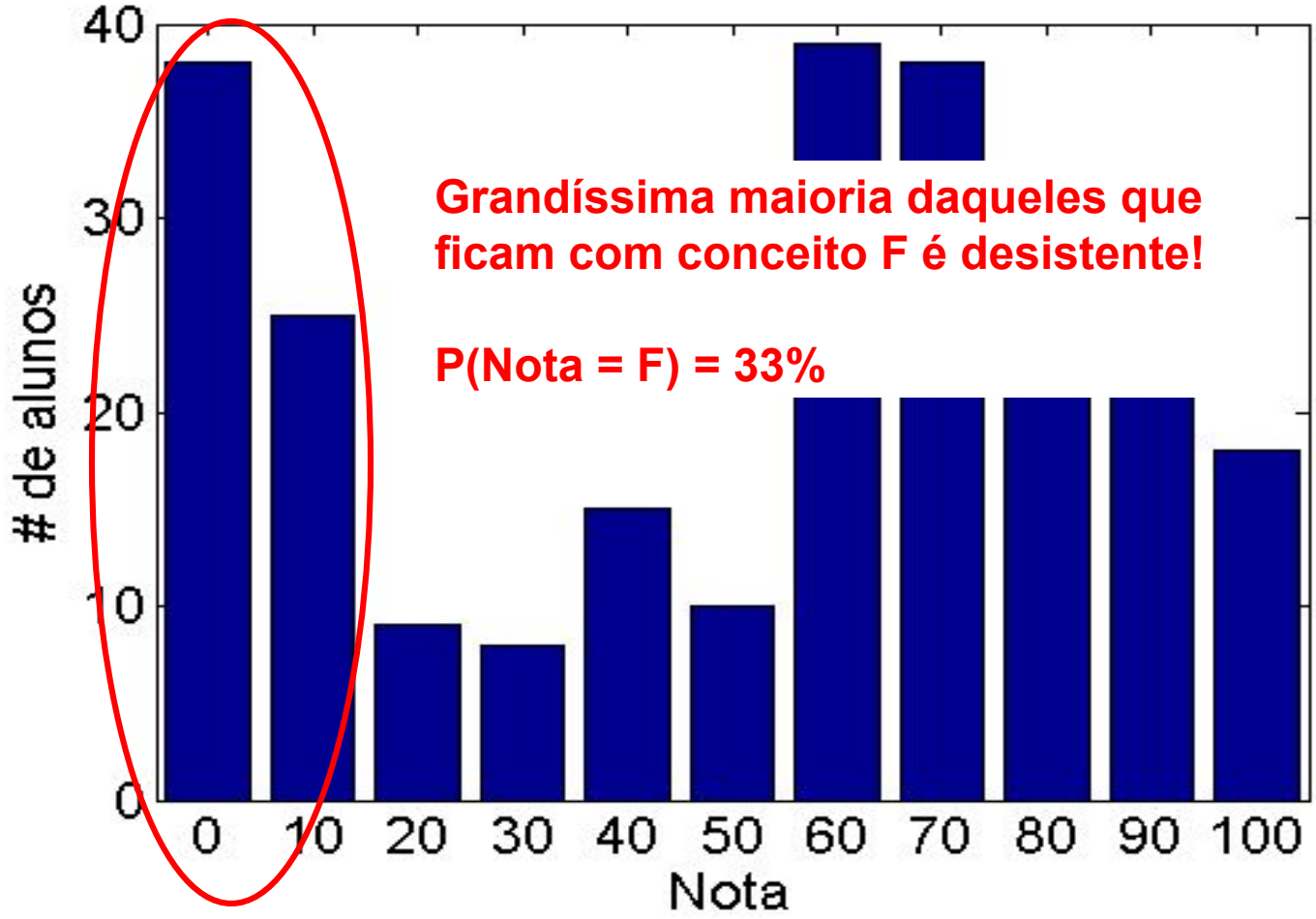
Observações

- Dedicação é fundamental
  - **Só se aprende a programar programando**
  - Trabalhos práticos requerem afinco e muitas horas de esforço
  - Provas e trabalhos exigindo conceitos + raciocínio lógico + análise de problemas

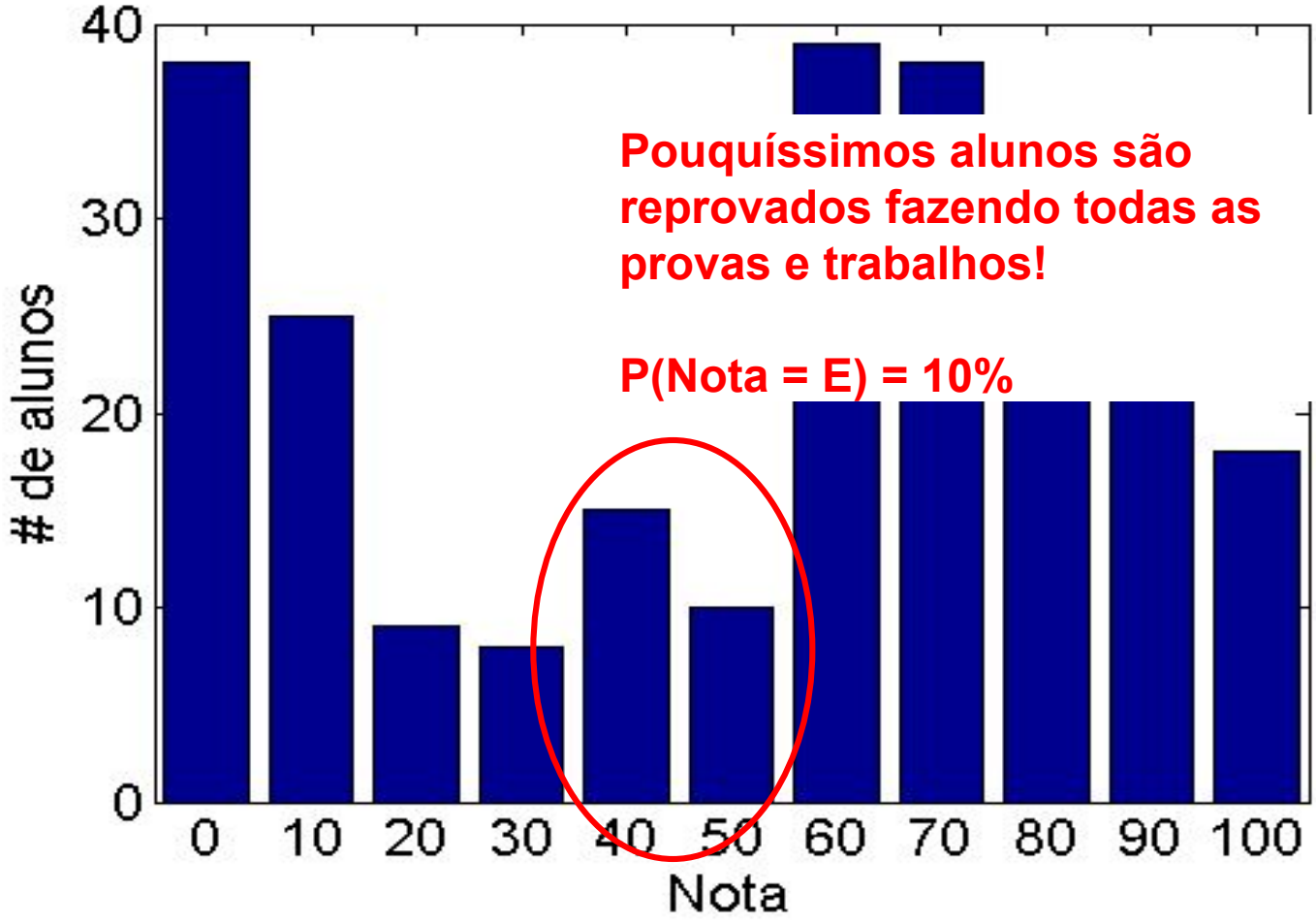
Desempenho dos Alunos



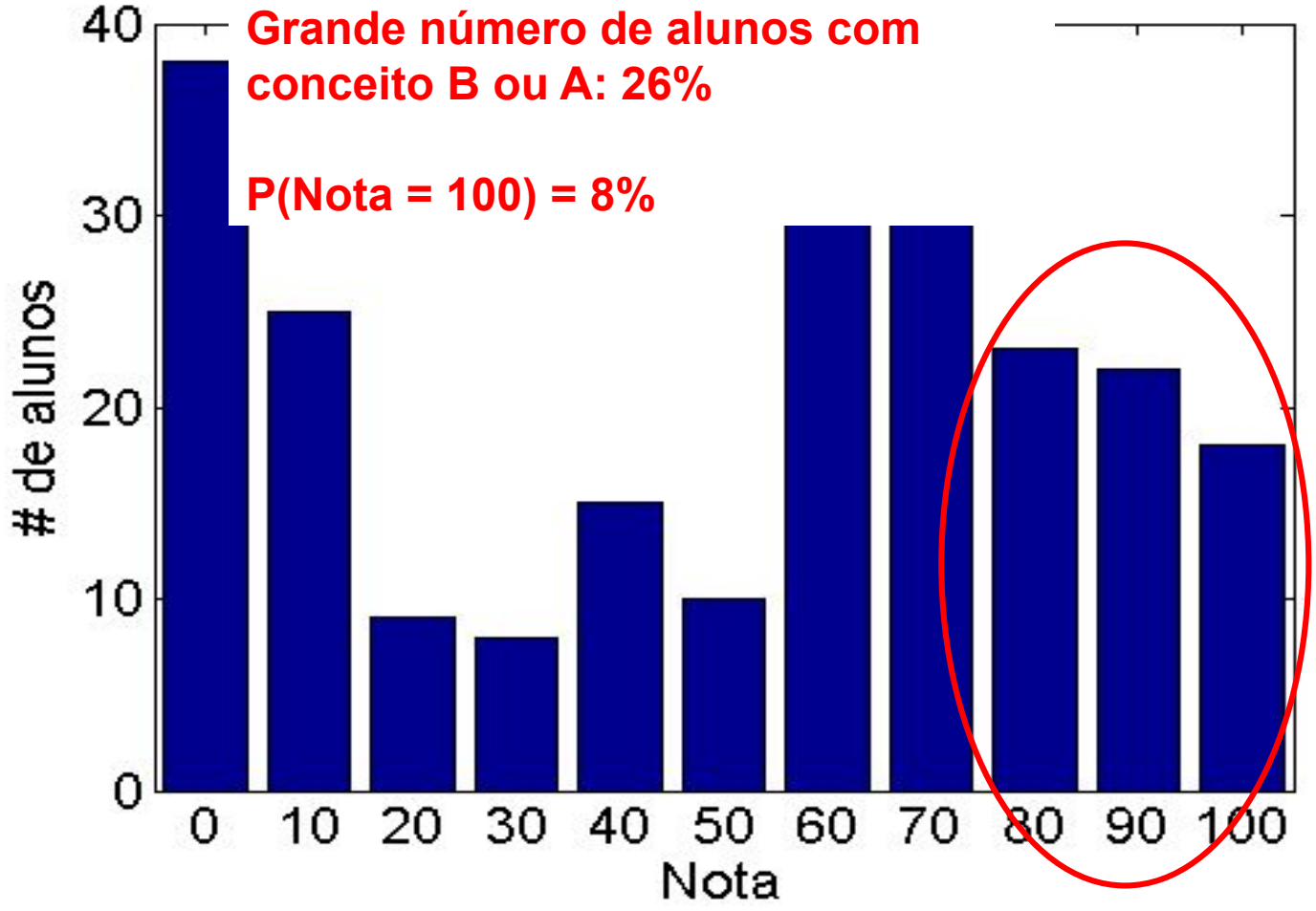
Desempenho dos Alunos



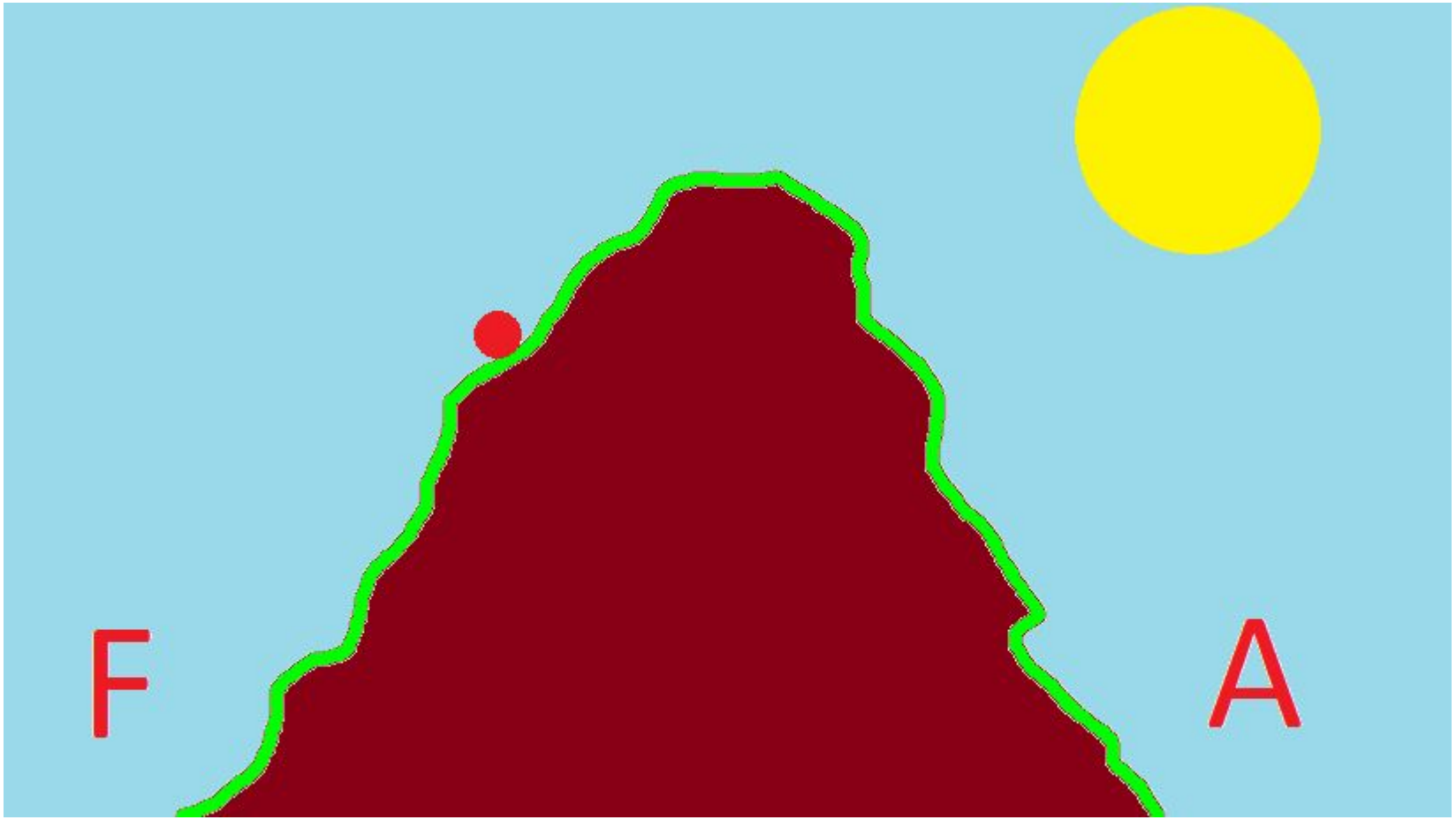
Desempenho dos Alunos



Desempenho dos Alunos



E você?



Observações

- Gostaria de aprovar todos, mas nem sempre isso é possível:
  - Programação é central em muitos cursos de exatas
  - Vai ser importante para o seu futuro profissional

Dicas

VIGORÔMETRO

Dicas

Tchaki Tchaki

Práticas

Dicas

Watermelon Sugar

Prova 1

Dicas

O Brasil tá lascado!

Prova 2

Dicas

Ai, Brasil

Prova 3

Dicas

Tô indignado!

TP

Dicas

- Muitos desistem pois não conseguem entender NADA que está sendo dado em aula
- CAUSA: Conteúdo é cumulativo, ou seja, para entender a aula x, é preciso ter entendido a aula x-1
- Solução: ir e prestar atenção
- Benefícios colaterais: menos estudo em casa, menos stress, menos fadiga

Dicas

Dicas

- Pergunte durante a aula!
  - Fico muito feliz com perguntas
  - Não existe pergunta boba
    - Eu **nunca** vou caçoar de uma pergunta feita por aluno
    - Então, não tenha medo de fazer perguntas bobas
  - A sua dúvida pode ser a dúvida de outro aluno

Dicas sobre o ERE

- É mais difícil verificar e acompanhar o aprendizado do aluno
- Aluno terá maior responsabilidade no processo de aprendizado
- Mais fácil trapacear e passar na disciplina
- Mais fácil não aprender
- Mais fácil atrasar a sua vida depois
- Semestre passará muito rápido, cuidado!

Sobre corrupção em sala de aula: cola, compra e cópia de trabalhos

- Cola
  - Não preciso pegar colando para identificar uma cola
  - Programas são como respostas dissertativas em português: estatisticamente, é impossível ter dois iguais!
  - Então, respostas estruturalmente iguais CARACTERIZAM uma cola

Sobre corrupção em sala de aula: cola, compra e cópia de trabalhos

- Compra e cópia de trabalhos
  - Será verificado na prova oral
  - Se você não conseguir responder sobre aspectos básicos do seu SUPOSTO programa, você será punido severamente

Perguntas?

- E-mail: olmo@dcc.ufmg.br
  - Leio somente e-mails de alunos com menos de 30 palavras e sem anexos
  - Outros vão direto para lixeira
- Respondo mensagens do chat do Teams rapidamente!
- Raramente respondo mensagens enviadas pelo Moodle!
- Podemos marcar um horário para atendimento individual
  - Melhor maneira de tirar dúvidas complexas
- Material da disciplina: www.dcc.ufmg.br/~olmo