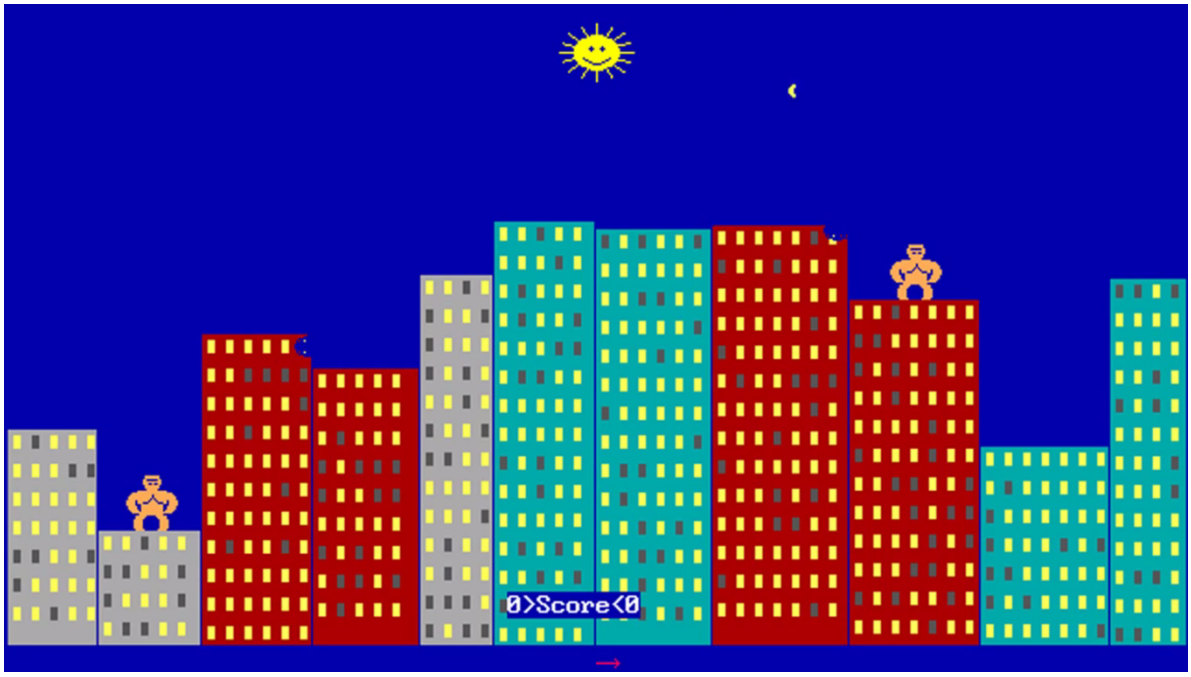
INSTITUTO SUPERIOR TÉCNICO

LICENCIATURA EM ENGENHARIA INFORMÁTICA E DE COMPUTADORES

PROJETO DE INTRODUÇÃO À ARQUITETURA DE COMPUTADORES

GORILAS



2018 / 2019

OBJETIVO

O projeto consiste no desenvolvimento de uma versão minimalista do jogo “Gorillas”. Neste jogo o objetivo é tentar acertar no gorila adversário atirando uma banana. O jogo decorre numa janela de texto (consola) onde vão sendo mostradas as animações das jogadas e o resultado.

Neste documento são descritos os detalhes de funcionamento pretendidos para o jogo. O jogo será programado usando a linguagem Assembly para o processador P3. O desenvolvimento e teste do programa serão realizados usando o simulador do P3 (p3sim).

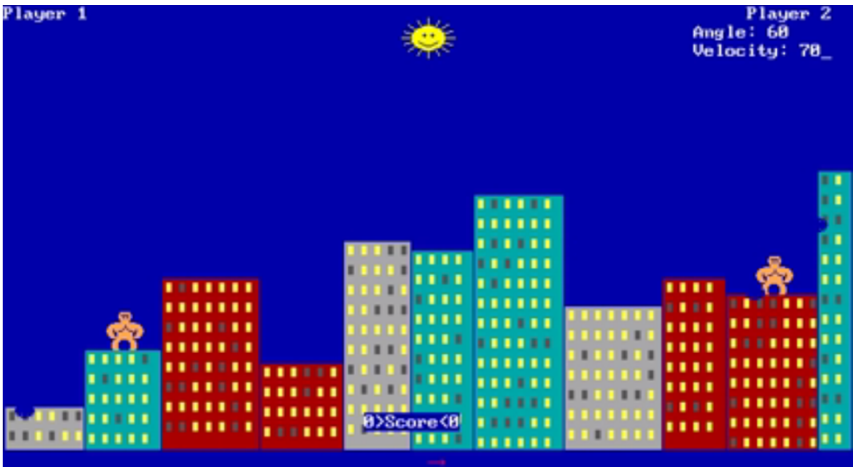
DESCRIÇÃO DO JOGO

O objetivo deste jogo é atingir o gorila adversário com uma banana atirada pelo jogador. Na versão original o jogo é destinado a 2 jogadores, onde cada um atira uma banana à vez. Cada gorila é colocado nos extremos do ecrã e são adicionados edifícios aleatoriamente ao cenário, dificultando o cálculo da trajetória da banana. Para aumentar a dificuldade, a velocidade do vento varia entre jogos.

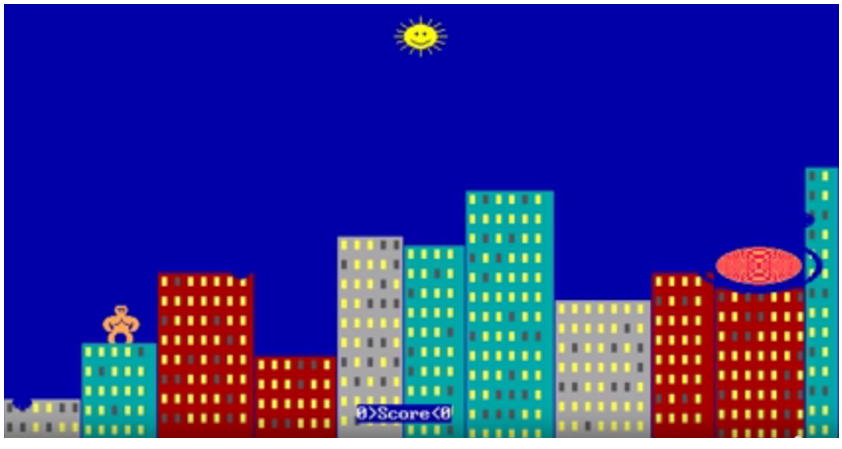
Para jogar, cada jogador indica a força e o ângulo com que atira a banana. Após cada jogada, o jogo mostra a trajetória da banana, simulando o efeito da gravidade. Um jogador ganha um ponto quando acertar no oponente.

EXEMPLO DA EXECUÇÃO DO JOGO

Introdução da jogada: ângulo = 60, velocidade = 70



Um gorila acerta com a banana no outro:



Mais detalhes sobre o jogo original e variantes podem ser encontrados aqui:

https://en.wikipedia.org/wiki/Gorillas_(video_game)

VERSÃO SIMPLIFICADA PARA O PROJETO

Para o projeto da cadeira, vamos simplificar o jogo das seguintes formas:

- não existem edifícios;

- não existe contribuição do vento;

- a interface é muito rudimentar, consistindo apenas num único caracter ASCII para a banana que é arremessada, e um conjunto de 5 a 10 caracteres ASCII para os gorilas (a escolha desses caracteres fica ao critério de cada grupo, podendo desta forma os alunos praticar um pouco de "ASCII Art").

Os restantes aspetos do jogo devem ser mantidos, nomeadamente:

- colocar os 2 gorilas no início do jogo em posições aleatórias (mas afastados entre si);

- permitir a introdução do ângulo e velocidade através do teclado numérico do P3;

- calcular o efeito da força da gravidade sobre o projétil;

Para simplificar, pode ser sempre o mesmo gorila (por exemplo, o da esquerda) a tentar acertar no adversário. Nesse caso a pontuação é apenas um número que contabiliza o número de jogadas com sucesso.

IMPLEMENTAÇÃO

A implementação está dividida em duas fases, com datas separadas para cada entrega.

PRIMEIRA FASE

A primeira fase consiste na implementação de uma funcionalidade que é crucial à implementação, que é o cálculo da trajetória do projétil.

Este cálculo tem como entrada a posição, ângulo, e velocidades iniciais. Dadas estas entradas, é possível calcular a posição em cada instante de tempo de acordo com as fórmulas indicadas em:

https://en.wikipedia.org/wiki/Projectile_motion#Displacement

Note que para este cálculo pode ser necessário usar funções trigonométricas, embora o processador P3 não tenha suporte para estas. Para obter estes valores deve procurar uma função que aproxime as funções pretendidas.

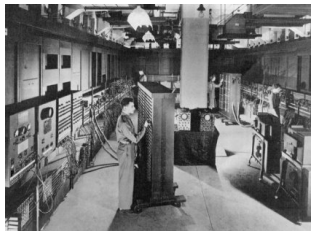
O cálculo deverá recorrer a uma representação em vírgula fixa, em que, num número, *n* bits são usados para a parte inteira (incluindo sinal, se tal for necessário), e os restantes *16-n* são usados para a parte fracionária.

Deve também entregar nesta fase um pequeno programa de teste em Assembly que demonstra o funcionamento correto deste código. Para simplificar, este programa pode fornecer a velocidade, ângulo e posição iniciais como constantes predefinidas no código Assembly, e mostrar a posição do projétil através de registos cujo conteúdo é analisado no simulador do P3.

Após esta parte estar a funcionar, devem também encapsular a implementação numa função (subrotina) que será depois útil para a segunda parte do projeto.

Curiosidade:

Em 1945, o ENIAC foi um dos primeiros computadores de uso genérico a ser desenvolvidos no Mundo, na universidade da Pensilvânia nos Estados Unidos. Este foi criado para calcular as tabelas com as trajetórias de artilharia, um cálculo semelhante ao que tem de ser efetuado neste trabalho.



SEGUNDA FASE

Nesta fase é construído o jogo, usando o código da primeira fase como parte da implementação.

Sugere-se o seguinte plano de desenvolvimento:

- O primeiro passo é pensar a estrutura geral da aplicação. A estrutura recomendada é a seguinte: após a inicialização do jogo, o código deverá entrar num ciclo infinito. No decorrer desse ciclo, poderão ocorrer várias interrupções, quer de dispositivos de E/S (por exemplo, do teclado numérico para inserir o ângulo e velocidade do lançamento), quer do temporizador (para marcar a passagem do tempo e proceder à atualização da posição do projétil seguida da deteção de colisão).

- Sugerimos que comece por esboçar esta estrutura (usando pseudocódigo, ou um diagrama como por exemplo um fluxograma) e discuti-la com o docente de laboratório. É importante que esta discussão ocorra pouco tempo após a primeira entrega.

- Implemente uma funcionalidade de cada vez, testando-a de forma exaustiva antes de passar à implementação da próxima funcionalidade.

- Note que em geral as interrupções devem ser rotinas muito curtas, normalmente limitando-se a ajustar o valor de uma ou mais variáveis, que depois serão utilizadas para controlar a execução do programa principal.

- Comente e indente devidamente o código desenvolvido.

VALORES ALEATÓRIOS

Quando inicia um novo jogo, é necessário obter um valor aleatório para determinar a posição dos gorilas. O Anexo A descreve um algoritmo para a geração de valores aleatórios de 16 bits.

DÚVIDAS NA ESPECIFICAÇÃO DO ENUNCIADO

Sempre que o enunciado não defina completamente como deve implementar alguma parte da solução, deve tomar a decisão que achar mais apropriada, tendo sempre como objetivo melhorar a experiência de jogo. Estas decisões devem ser descritas e justificadas no relatório final.

ENTREGAS

DIA 26 DE OUTUBRO, 23:59

Código (ficheiro no formato .as)

DIA 23 DE NOVEMBRO, 23:59

Código + relatório (ficheiro no formato .as + ficheiro no formato .pdf)

MATRIZ DE AVALIAÇÃO

Relatório: 20%

Será avaliada a capacidade síntese, a qualidade da escrita, e capacidade de apresentar de forma direta, mas exaustiva as decisões mais importantes que foram tomadas na conceção do programa.

Código: 40%

Será avaliada a qualidade do código, nomeadamente no que diz respeito à sua modularidade (e.g., uso de funções reutilizáveis e fáceis de manter), à passagem de parâmetros pela pilha de forma que obedeça às convenções ensinadas na cadeira, ou à utilidade dos comentários, ausência de rotinas de tratamento de interrupções longas, entre outras.

Funcionalidade/Visualização: 40%

Este item avalia o funcionamento correto do programa.

ANEXO A – GERAÇÃO DE SEQUÊNCIA PSEUDOALEATÓRIA

O seguinte algoritmo gera uma sequência N0, N1, N2, ..., Ni, Ni+1, ... aparentemente aleatória de números de 16 bits:

```
Máscara = 1000 0000 0001 0110 b;
i = 0;
while (true) {
  if (Ni_0 == 0) /* Testa o bit de menor peso de Ni */
    Ni+1 = rotate_right (Ni);
  else
    Ni+1 = rotate_right (XOR (Ni, Mascara));
  i++
}
```

Em cada iteração do ciclo lê-se o valor mais recentemente gerado Ni (começando a partir do N0 na primeira vez que é invocada) e gera-se um novo valor Ni+1. Para mudar a sequência entre execuções do programa, a semente desta sequência (N0 ≠0) deve ser obtida a partir de um parâmetro que varie de execução para execução (por exemplo, na inicialização pode-se pedir para premir uma tecla para começar o jogo, e contar o número de instruções executadas até tal acontecer). Para obter um valor aleatório entre 0 e M-1 pode dividir o valor de Ni por M e usar como número aleatório o resto, mas sempre sem modificar o valor de Ni.