ESCOLA
SUPERIOR
DE MEDIA
ARTES
E DESIGN

P.PORTO

Pang

Animação Gráfica
Docente: Prof. Bruno Oliveira

Licenciatura em Tecnologias e Sistemas de Informação para Web

Bernardo Ferreira, 9170125

Carlos Guedes, 9170138

Rodrigo Queirós, 9170312

Vila do Conde, janeiro 2019

Resumo

No âmbito da unidade curricular de Animação Gráfica, da licenciatura em Tecnologias e Sistemas de Informação para Web, fomos propostos a desenvolver em JavaScript o jogo Pang, um classico lançado em 1989 por Mitchell Corporation.

Palavras-chave: Animação Gráfica, JavaScript, Pang

Abstract

Within the curricular unit of Graphic Animation, the degree in Technologies and Information Systems for Web, we were proposed to develop in JavaScript the game Pang, a classic released in 1989 by Mitchell Corporation.

Keywords: Graphic Animation, JavaScript, Pang

Lista de Siglas

JavaScript – JS

Escola Superior Media Artes e Design – ESMAD

Hypertext Markup Language – HTML

Cascade Style Sheet – CSS

Animação Gráfica – AG

Sumário

0 - Introdução

O principal objetivo deste trabalho é recriar o jogo Pang, um jogo de arcade também conhecido por Pomping World e lançado em 1989, pretendemos também que a nossa versão deste jogo tenha a maior qualidade possível e que no final seja divertido jogar o nosso próprio jogo.(Pang *(video game)*, 2018)

Para isso começamos por uma análise dos requisitos e do jogo em si, a seguir fizemos uma reunião para discutir objetivos e formas de abordagem do que precisamos de fazer, marcando também o início do seu desenvolvimento. Estando a fase de análise e planeamento feitas, passamos então ao desenvolvimento onde muitas vezes juntos nos sentamos a resolver o trabalho. A última fase foi a de correção de erros, testes, últimos retoques e a realização deste relatório com as conclusões do projeto.

O presente relatório encontra-se dividido em 4 partes. Na primeira parte apresentamos como abordamos o trabalho, a seguir fazemos uma análise do algoritmo do trabalho e mostramos a nossa forma de o resolver, numa terceira parte falamos dos problemas encontrados e como os solucionamos, por último, apresentamos as conclusões do projeto.

1 – Abordagem

Para a realização do jogo tivemos de seguir certas regras:

- O jogo deve ter pelo menos 3 níveis
- Os gráficos devem ser criados por nos
- O jogo pode ser no máximo de 2 jogadores
- Cada jogador só pode ter um arpão

Tendo estas regras em conta e tendo experimentado o jogo, começamos a planear o que íamos implementar e quais iam ser as nossas diferenças.

Reunimo-nos várias vezes para planear e desenvolver o jogo em conjunto.

Para resolver os problemas que fomos encontrados ao longo do projeto a nossa solução foi sempre pesquisar, reunir a equipa e tentativa erro. Não encontramos assim tantos obstáculos porque os métodos aprendidos em aula eram perfeitos para a implementação deste trabalho.(Oliveira, [s.d.])

2 – Algoritmo

Começamos o desenvolvimento do jogo pela criação do player, a seguir do arpão e as bolas.

Para os 3 a técnica usada foi a mesma, funções construtoras com as funções lá dentro draw e update.

```
function Player(image, playerWidth, playerHeight, step, spriteLine) {

  this.image = image
  this.playerWidth = playerWidth
  this.playerHeight = playerHeight
  this.step = step
  this.stepUpDown = step
  this.spriteLine = spriteLine
  this.image.src = "images2/gb_walk.png"
```

Depois para criar por exemplo o player tínhamos de criar o objeto. No caso das bolas e arpões e mais tarde powerups também colocávamos dentro de um array.

```
//Creation of player
let playerWidth = 1000 / 9
let playerHeight = 550 / 5
let step = 0
let spriteLine = 0
let player1 = new Player(new Image(), playerWidth, playerHeight, step, spriteLine);
```

O draw desenhava o objeto e o update atualizava os valores de posição. No update também eram feitas algumas colisões que resultavam na mudança de direção

```
this.draw = function () {
  ctx.beginPath();
  ctx.arc(this.x, this.y, this.radius, 0, 2 * Math.PI);
  ctx.fillStyle = 'red'
  ctx.fill();
}

this.update = function () {
  this.x += this.vx
  this.y += this.vy
  //console.log("vy: " + this.vy)
  if (this.y + this.radius >= canvas.height) {
    this.vy = -this.vy
    //this.y = canvas.height- this.radius
    //this.vy = (this.velIn + 20) * Math.sin(this.ang * Math.PI / 180)
  }
```

Para o movimento do player usamos variáveis bool ao carregar nas teclas pretendidas e passamos para o player a partir de uma função que fica a escuta.

```
function keyUp(e) {
  switch (e.keyCode) {
    case 39:
      right = false
      break;
    case 37:
      left = false
      break;
    case 38:
      up = false
      break;
    case 40:
      down = false
      break;
```

```
//ListenEvent and Draw player mov
if (space) {
  player1.listenEvent(false, false, false, currentFrame, false, false, true)
}
else {
  if (right) {
    player1.listenEvent(true, false, false, currentFrame, false, false, false)
  }
  else if (left) {
    player1.listenEvent(false, true, false, currentFrame, false, false, false)
  }
  else if (up) {
    player1.listenEvent(false, false, false, currentFrame, true, false, false)
  }
  else if (down) {
    player1.listenEvent(false, false, false, currentFrame, false, true, false)
  }
  else {
    player1.listenEvent(false, false, true, currentFrame, false, false, false)
  }
}
```

Como as variáveis dentro das funções construtoras não são globais precisamos de criar uma função que dê return as variáveis pretendidas

```
this.getCurrentPos = function () {

  let ballPos = { x: this.x, y: this.y, r: this.radius }
  return ballPos

}
```

Os powerups foram desenvolvidos a partir de id randoms e probabilidade random

```
randPUP = Math.floor(Math.random() * 5)

if (randPUP == 1) {
  let x = balls[q].getCurrentPos().x - 25
  let y = balls[q].getCurrentPos().y - 25
  let id = Math.floor(Math.random() * 4) + 1
  let img = new Image()
  powerups.push(new PowerUp(x, y, id, img))
```

Temos 4 powerups, um que congela a posição das bolas temporariamente, um que modifica o máximo número de arpões para 5 temporariamente, um que faz com que o arpão fique preso no topo e um que nos dá mais uma vida

Dentro do construtor era atribuído a imagem certa e a partir do id era ativado o powerup certo

```
switch (this.id) {
  case 1:
    this.img.src = "images2/powerup1.png"
    break;
  case 2:
    this.img.src = "images2/powerup2.png"
    break;
  case 3:
    this.img.src = "images2/powerup3.png"
    break;
  case 4:
    this.img.src = "images2/powerup4.png"
    break;
  default:
    console.log("Error 404: PowerUp not found")
    break;
}
```

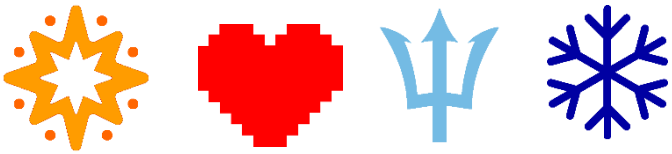
```
function powerupActivate(i) {

  switch (powerups[i].getCurrentPos().id) {
    case 1: //Harpon stick on top
      powerup1 = true
      break;
    case 2: //Harpon unlimited
      powerup2 = true
      maxHarpoons = 5
      // times up maxharpoons = 1
      break;
    case 3: //Freeze
      powerup3 = true
      break;
    case 4:
      if (lives < 5) { lives += 1 }
      break;
    default: console.log("Error on power up power id detect")
      break;
  }
```

Alguns powerups podiam receber as mudanças na função da imagem anterior outros precisavam de por exemplo limitar o update das bolas quando ativo

```
//Freeze
if (powerup3 == false) {

  balls[q].update()
```



Para os powerups estarem completos só falta as suas colisões, aparecerem na destruição de uma bola, descer do ponto inicial até ao chão e um timer

```
for (let i = 0; i < powerups.length; i++) {
  powerups[i].update()

  if (powerups[i].getCurrentPos().x <= player1.getCurrentPos().x + playerWidth &&
    powerups[i].getCurrentPos().x + 50 >= player1.getCurrentPos().x &&
    powerups[i].getCurrentPos().y <= player1.getCurrentPos().y + playerHeight &&
    powerups[i].getCurrentPos().y + 50 >= player1.getCurrentPos().y
  ) {
    console.log("Powerup activate")
    powerupActivate(i)
    powerups.splice(i, 1)
  }
```

```
this.update = function () {
  //Powerup - Ground
  if (this.y + 50 > canvas.height) {
    this.cy = 0
    this.y = canvas.height - 50
  }
  this.y += this.cy

}
```

Em relação ao timer é guardado o frame no momento em que é ativo e fica a ser comparado com o valor de 500 para depois ser desativado

```
currentFrame++
if (currentFrame >= 500) {
  currentFrame = 0
  powerup3 = false
}
```

Para as colisões de bolas, player, arpões, powerup e plataformas foi usada sempre a mesma estratégia

```
if (balls[q].getCurrentPos().x + balls[q].getCurrentPos().r >= player1.getCurrentPos().x
  && balls[q].getCurrentPos().x - balls[q].getCurrentPos().r <= player1.getCurrentPos().x + playerWidth
  && balls[q].getCurrentPos().y + balls[q].getCurrentPos().r >= player1.getCurrentPos().y
) {
```

Desenvolvemos também um score, que a pontuação máxima só é obtida se não perdemos nenhuma vida visto que o scoreMultiplier aumenta ao acertar a bola e volta ao seu valor inicial ao ser acertado

```
//Collision ball and harpoon
scoreMultiplier += 0.1
currentScore = parseInt(currentScore + (scorePlus * scoreMultiplier))
```

```
//Collision ball and player
scoreMultiplier = 1
```

Ao perder ou ganhar o currentScore é comparado com o currentBest para determinar se o valor é guardado ou não

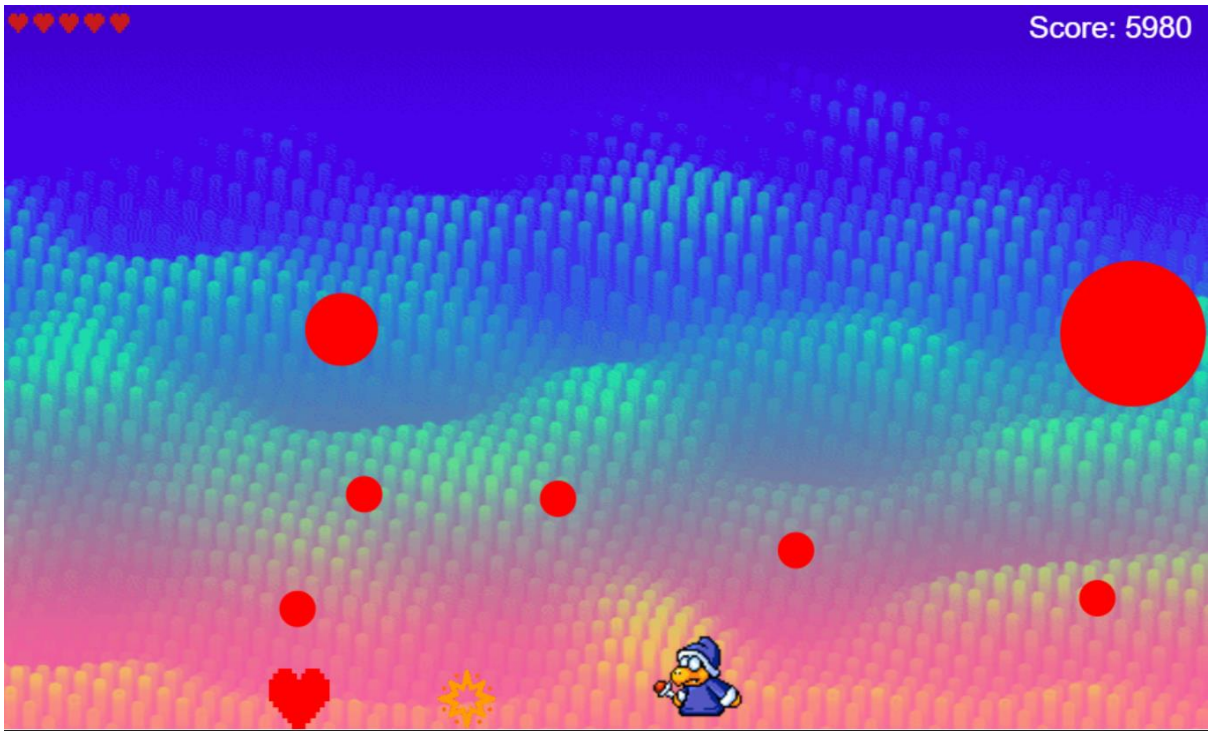
```
if (lives == 0) {
  gameOverBool = true
  if(currentScore > localStorage.getItem("currentBest")){
    currentBest = currentScore
    localStorage.setItem("currentBest", currentBest)
  }
  gameOver()
}
```

No final só faltava um menu onde pudéssemos iniciar o jogo, ver os controles, escolher os níveis a jogar, ver o highscore e por o jogo em pausa


```
function menu() {
  scoreMultiplier = 1
  currentScore = 0
  //Menu
  menuBool = true
  levelsMenuBool = false
  controlsBool = false
  ctx.fillStyle = "black"
  ctx.rect(0, 0, canvas.width, canvas.height)
  ctx.fill()
  let img = new Image()
  img.src = "images2/pang.png" //https://i.imgur.com/7Gc8NXt.png
  img.addEventListener("load", function () {
    ctx.drawImage(img, 300, 0, 400, 200)
  })

  ctx.font = "50px Arial"
  ctx.fillStyle = "white"
  let text = "Start Game"
  let textWidth = ctx.measureText(text).width
  ctx.fillText(text, (canvas.width / 2) - (textWidth / 2), 275)
```

Sendo este o resultado



3 – Dificuldades encontradas

A primeira dificuldade encontrada foi o sprite do player, para resolver o problema tivemos que modificar a imagem, calcular espaços e tentar que o movimento ficasse o mais perfeito possível.

Fazer com que tudo ficasse funcional e a trabalhar direito tendo em conta o design também foi um dos nossos maiores problemas, foram várias as soluções dependendo do problema em questão.

(*RPG Maker MV RPG Maker VX Super Nintendo Entertainment System Mario Sprite, mario PNG clipart | free cliparts | UlHere*, [s.d.])

Mesmo tendo começado o desenvolvimento do jogo cedo, os problemas que foram surgindo ao longo do tempo fizeram com que a sua conclusão se atrasasse tendo em conta também outros trabalhos e testes pelo meio.

O nosso maior problema foi se calhar as colisões que falhavam 5% das vezes, a nossa solução foi tentar melhorar as fórmulas o mais possível, mas problemas como frama skip que não podíamos controlar podem sempre acontecer.

Conclusão

Com este trabalho conseguimos recriar o jogo Pang em JS e Canvas, conseguimos implementar a maior parte do que pretendíamos e por isso concluímos que o resultado foi um sucesso, mas que ainda pode ser melhorado.

Conseguimos resolver todos os pontos pretendidos pelo trabalho e ainda fazer mais, mas encontramos alguns obstáculos que não conseguimos ultrapassar, damos o exemplo das plataformas.

As plataformas foram desenvolvidas, mas devido a problemas em gerir todas as colisões acabamos por não implementar, deixando em comentário no código. Para uma continuação deste trabalho sugerimos a melhoria do movimento do player e da implementação das plataformas com o problema de colisões resolvido. Para além disso sugerimos o desenvolvimento de mais níveis e a implementação de uma arte mais uniforme.

Neste trabalho desenvolvemos a maior parte dos gráficos do jogo, sendo que os que não criamos estão referenciados propriamente.

Os pontos positivos deste trabalho foram vários, conseguimos aprofundar conhecimentos e melhorar técnicas de programação, conseguimos realizar o jogo com um bom resultado e na nossa opinião a nossa equipa de trabalho resultou muito bem.

Concluímos, portanto, que mesmo tendo encontrado obstáculos conseguimos juntos chegar a um bom resultado e estamos orgulhosos do que fizemos e que aprendemos bastante e sentimos que este trabalho foi um passo importante para o nosso desenvolvimento profissional.

Referências Bibliográficas

Background 3 - GIF background 3d design - animated GIF on GIFER - by Ananadar - [Em linha] [Consult. 6 jan. 2019]. Disponível em WWW:<URL:https://gifer.com/en/iCg>.

Background 1 - GIPHY - Background GIF - Find & Share on GIPHY, [s.d.]. [Consult. 6 jan. 2019]. Disponível em WWW:<URL:https://media.giphy.com/media/BHNfhgU63qrks/giphy.gif>.

Background 2 - GIPHY - Background GIF by GIPHY CAM - Find & Share on GIPHY, [s.d.]. [Consult. 6 jan. 2019]. Disponível em WWW:<URL:https://media.giphy.com/media/l378wcSfS7eXWQgla/giphy.gif>.

OLIVEIRA, Bruno - Textbook Animação Gráfica 18/19. [s.d.]) 57.

Pang Logo - Logo Vector Online 2019 - , [s.d.]. [Consult. 6 jan. 2019]. Disponível em WWW:<URL:https://www.finam.club/pang-logo.html>.

Pang (video game) - Em Wikipedia [Em linha] [Consult. 6 jan. 2019]. Disponível em WWW:<URL:https://en.wikipedia.org/w/index.php?title=Pang_(video_game)&oldid=864554389>.

Player sprite - RPG Maker MV RPG Maker VX Super Nintendo Entertainment System Mario Sprite, mario PNG clipart | free cliparts | UIHere - [Em linha] [Consult. 6 jan. 2019]. Disponível em WWW:<URL:https://www.uihere.com/free-cliparts/rpg-maker-mv-rpg-maker-vx-super-nintendo-entertainment-system-mario-sprite-mario-6442776>.