



LICENCIATURA EM ENGENHARIA
INFORMÁTICA

Trabalho Prático 1 - Breakout

Tecnologias Multimédia

2022/2023

Autor:

Diogo Bernardes, 27984

Ricardo Conde, 23452 *Orientador:*

Professor Marcelo Fernandes

09 de Abril de 2023

Índice

1	Introdução	2
2	Jogo	3
2.1	Apresentação do jogo	3
2.2	Cenas	3
2.2.1	Menu	3
2.2.2	Primeiro nível - Jogo	4
2.2.3	Segundo nível - Jogo	4
2.2.4	Bonus - Jogo	5
2.2.5	GameOver / Congratulations - Jogo	5
2.3	GodMode	6
2.4	Bugs / Futuras implementações	6
3	Implementação	7
3.1	Desenvolvimento do Jogo	7
3.1.1	Variáveis	7
3.1.2	Preload	8
3.1.3	Create	8
3.1.4	Update	9
3.1.5	BounceOfPaddle	10
3.1.6	CreateBricks	10
3.1.7	Manage e BallHitBrick	11
3.1.8	Informations e hitLava	11
3.1.9	Create Bonus	12
3.1.10	Positive Bonus	13
3.1.11	Negative Bonus	13
3.1.12	HandleInput e SetupInput	14
4	Conclusão	15
5	Links	15

1 Introdução

O presente relatório tem como propósito desenvolver um jogo em Phaser3 proposto na Unidade Curricular de Tecnologias Multimédia, com o intuito de aplicarmos nele toda a matéria aprendida no decorrer do semestre.

O jogo escolhido para este trabalho foi o Breakout, e no decorrer deste relatório iremos tentar abordar todo o processo de desenvolvimento do mesmo.

Para além disto vamos também falar acerca das dificuldades obtidas e de algumas melhorias que poderão ser implementadas futuramente.

2 Jogo

2.1 Apresentação do jogo

O jogo Breakout consiste numa camada de tijolos que é alinhada no topo da tela, onde a bola passa pela tela, batendo nas paredes, laterais e superiores, da tela. Quando um tijolo é atingido pela bola, esta rebate de volta e o tijolo é destruído. O jogador perde uma vida quando a bola toca na parte inferior da tela. Para prevenir que isso aconteça, o jogador move o “paddle” para rebater a bola para cima, mantendo-a no jogo.



Figura 1: Breakout

2.2 Cenas

2.2.1 Menu

A primeira página desenvolvida para este jogo, foi uma espécie de menu, que será a primeira página a ser apresentada ao utilizador. Este menu apenas possui uma imagem de fundo com o título do jogo e um botão "Start game", que quando clicado dá início ao jogo.

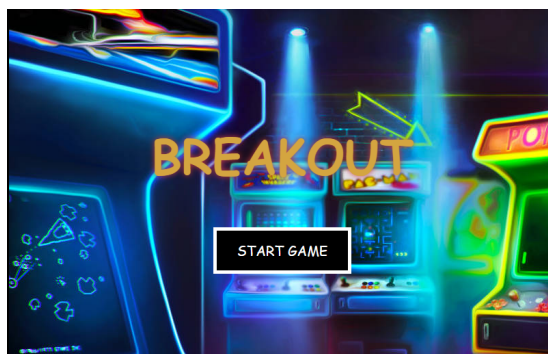


Figura 2: Menu

2.2.2 Primeiro nível - Jogo

Após termos clicado no botão do menu, somos redirecionados para a primeira cena em si do jogo. Está primeira cena é referente ao primeiro nível do jogo, onde são apresentados os blocos que o jogador terá de destruir, a bola e o paddle para que consigamos destruir os blocos e as informações necessárias para o jogador como as vidas restantes e a sua pontuação atual. Para além disto, quando os blocos são destruídos existe uma pequena probabilidade de estes darem um bônus ao jogador podendo este ser bom ou mau.



Figura 3: Primeiro Nível - Jogo

2.2.3 Segundo nível - Jogo

Quando o jogador destrói todos os blocos existentes no primeiro nível é automaticamente redirecionado para o segundo nível, nível esse que possui os mesmos assets que o primeiro, no entanto é elevada a dificuldade do mesmo com o aumento da velocidade da bola e mais blocos.



Figura 4: Segundo Nível - Jogo

2.2.4 Bonus - Jogo

Quando o jogador destrói um bloco existe a probabilidade de este lhe dar um bônus podendo este ser bom ou mau. Quando o bônus é bom o bloco deixa cair uma laranja após ser destruído, quando o bônus é negativo, o bloco deixa cair uma maçã podre.

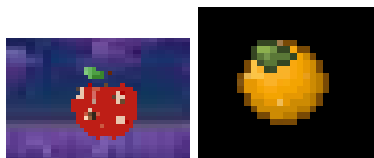


Figura 5: Bônus - Jogo

2.2.5 GameOver / Congratulations - Jogo

Quando o jogador perde todas as vidas ou quando o jogador finaliza o jogo, o jogo é dado por terminado e é enviada uma mensagem informativa ao jogador com a sua pontuação final e com a indicação de vitória ou derrota.



Figura 6: GameOver - Jogo

2.3 GodMode

Tal como foi proposto pelo professor da unidade curricular, foram realizados alguns "cheats" para este jogo, para que assim fosse mais simples testar o mesmo posteriormente. Assim sendo os "cheats" implementados foram:

- Quando a tecla **N** é pressionada, o tamanho do paddle é aumentado em 2.5.
- Quando a tecla **L** é pressionada, as vidas são incrementadas em 1 por cada clique.
- Quando a tecla **B** é pressionada, tanto o tamanho do paddle como as vidas são resetadas.
- Quando a tecla **Z** é pressionada, o jogo passa para o nível 2.

```
function godMode() {  
  // Define as variáveis iniciais  
  var maxLives = 3;  
  
  // Adicione um listener para a tecla "N"  
  this.input.keyboard.on("keydown-N", function () {  
    paddle.setScale(2.5); // aumenta o tamanho do paddle em 2 vezes  
  });  
  
  // Adicione um listener para a tecla "L"  
  this.input.keyboard.on("keydown-L", function () {  
    lives += 1; // aumenta o número de vidas em 1  
    livesText.setText("Lives: " + lives); // atualiza o texto das vidas  
  });  
  
  // Adicione um listener para a tecla "B"  
  this.input.keyboard.on("keydown-B", function () {  
    paddle.setScale(1); // volta o tamanho do paddle ao normal  
    lives = maxLives; // volta o número de vidas para 3  
    livesText.setText("Lives: " + lives); // atualiza o texto das vidas  
  });  
  
  // Adicione um listener para a tecla "Z"  
  this.input.keyboard.on("keydown-Z", function () {  
    level = 2;  
    createBricks2();  
  });  
}
```

Figura 7: GodMode

2.4 Bugs / Futuras implementações

Na realização deste projeto a maior dificuldade que encontramos foi com a passagem de níveis, pois a partir do momento que o primeiro nível é finalizado, no nível seguinte parece que os blocos do nível 1 ficam sobrepostos aos do nível 2 embora fiquem invisíveis.

Apesar de muitas tentativas para contornar este problema, este sempre persistiu a todas as diferentes implementações que fomos implementando. A única maneira que arranjei para que o jogo possa terminar com uma vitória foi definir uma pontuação em que no caso do jogador conseguir atingir a mesma ele ganha o jogo. Embora eu saiba que está forma de atribuir a vitória ao jogador não seja a forma correta, foi a única que encontrei até ao momento da entrega.

Assim sendo, para trabalho futuro espero conseguir resolver este problema das passagens de níveis para que o jogo se comporte da maneira adequada ao mesmo.

3 Implementação

3.1 Desenvolvimento do Jogo

3.1.1 Variáveis

Inicialmente, começamos por declarar todas as variáveis necessárias para o funcionamento do jogo. Começamos por declarar as variáveis que serão utilizadas para os objetos, seguidamente declaramos as variáveis utilizadas para os sons, depois as variáveis de informação e as referentes aos bônus e por último as variáveis que armazenam a quantidade de blocos em cada nível. O jogador começa sempre com 3 vidas (“var lives = 3”) e com a sua pontuação em 0 (“var score = 0”). Todos os níveis irão ter bônus, podendo este ser negativo ou positivo.

```
var game = new Phaser.Game(config);

var ball;
var paddle;
var heart;
var bonus;

var warning;
var pBuff;
var sound;
var life;

var lives = 3;
var score = 0;
var livesText;
var scoreText;
var color = ["0xffffffff", "0xff0000", "0x00ff00", "0x0000ff"];
var level = 1;

var bonusCount = 0;
var bonusGoodActive = false;
var bonusBadActive = false;
var extraBallActive = false;

var bricks_level1 = 47;
var bricks_level2 = 64;
```

Figura 8: Variáveis

3.1.2 Preload

De seguida, são carregados todos os recursos de som e imagem que serão utilizados no decorrer do jogo.

```
function preload() {  
  this.load.image("paddle", "assets/images/paddle.png");  
  this.load.image("ball", "assets/images/ball.png");  
  this.load.image("background", "assets/images/background.png");  
  this.load.image("heart", "assets/images/heart.png");  
  this.load.image("bonusPositive", "assets/images/laranja.png");  
  this.load.image("bonusNegative", "assets/images/maça.png");  
  this.load.audio("destroyBrick", "assets/sounds/brickDestroy.mp3");  
  this.load.audio("newLife", "assets/sounds/newLife.mp3");  
  this.load.audio("danger", "assets/sounds/warning.mp3");  
  this.load.audio("buff", "assets/sounds/positiveBuff.mp3");  
}
```

Figura 9: preload

3.1.3 Create

A função “create()” cria a os elementos do jogo, adiciona os sons, define as propriedades do paddle e da bola e os respectivos collider’s, adiciona a “lava” ao limite inferior do jogo e invoca as funções que serão responsáveis por mostrar as informações do jogo, criar os blocos para o nível 1, utilizar o godMode e por fim a função responsável por criar os eventos para mover o paddle com o teclado ou com o rato.

```
function create() {  
  scene = this;  
  scene.add.image(400, 245, "background");  
  scene.scale.setDimensions(800, 400);  
  
  // Adicionar sons  
  sound = this.sound.add("destroyBrick");  
  life = this.sound.add("newLife");  
  warning = this.sound.add("danger");  
  buff = this.sound.add("buff");  
  
  // Código paddle  
  paddle = scene.physics.add.sprite(400, 490, "paddle");  
  paddle.setAngle(0);  
  paddle.body.setSize(150, 40);  
  paddle.setCollideOnBounds(true);  
  paddle.body.immovable = true;  
  
  // Código bola  
  ball = scene.physics.add.sprite(400, 300, "ball");  
  ball.setScale(0.60);  
  ball.body.setSize(120, 120);  
  ball.setCollideOnBounds(true);  
  ballGroup = this.physics.add.group();  
  ball.body.setCollideOnBounds = true;  
  ball.body.bounce.y = 1; //Salto bola vertical e horizontal  
  ball.body.bounce.x = 1;  
  function startBall() { // Tem a velocidade de 300  
    ball.setVelocity(300, 300);  
  }  
  
  //Delay para a bola começar  
  setTimeout(startBall, 500);  
  scene.physics.add.collider(ball, paddle, bounceOffPaddle);  
  
  //Define a variavel lava como um retangulo que ocupa toda a largura da tela e tem 10px de altura e de faz o collider com a bola  
  lava = scene.add.rectangle(0, 390, 800, 10, {x:0, y:390});  
  scene.physics.add.existing(lava);  
  lava.body.immovable = true;  
  scene.physics.add.collider(ball, lava, 0, Hittava);  
  
  //Chamar funções  
  infoTime();  
  createBrick();  
  godMode.call(this);  
  setupInput();  
}
```

Figura 10: Create

3.1.4 Update

Nesta função “Update” é verificado se o número de vidas do jogador é igual a zero. Se for esse o caso, significa que o jogador perdeu e a função exibe a mensagem "Game Over" em texto na tela e também exibe a pontuação do jogador. Em seguida, a função adiciona um botão de “restart” para que o jogador possa iniciar o jogo novamente. Quando o jogador clica no botão, a página é recarregada e o jogador é redirecionado para a página inicial. Além disso, a função pausa o jogo e desabilita as interações do jogador com o jogo, impedindo que ele continue interagindo após o fim do jogo.

Caso o jogador ainda tenha vidas, a função verifica se a variável que contém o número de blocos do nível 1 é igual a 0, no caso disso acontecer, significa que o jogador terminou o nível e automaticamente é redirecionado para o segundo nível e a variável "level" igualada a 2. Para além desta verificação, a função verifica também se a variável "level" é igual a 2 e se a pontuação do jogador é igual a 210, caso ambas as verificações sejam verdade, significa que o jogador terminou o nível e é exibida a mensagem "Congratulations!" em texto na tela e também exibe a pontuação do jogador.

```
function update() {
  if (lives === 0) {
    var gameOverText = this.add.text(
      game.config.width / 2,
      game.config.height / 3,
      "Game Over! Your score was: " + score,
      { font: "32px Arial", fill: "ffffff" }
    );
    gameOverText.setOrigin(0.5);

    var restartButton = this.add.text(
      game.config.width / 2,
      game.config.height / 3 + 150,
      "Jogar Novamente!",
      {
        font: "32px Arial",
        fill: "ffffff",
        stroke: "ffffff",
        backgroundColor: "#8878C3",
      }
    );
    restartButton.setOrigin(0.5);

    restartButton.setInteractive(); // Habilita a interatividade do botão
    restartButton.on("pointerup", function () {
      window.location.href = "index.html";
    });

    this.physics.pause();
    this.input.mouse.disableContextMenu();
    this.input.keyboard.enabled = false;
  }
  if (bricks_level1 === 0) {
    level = 2;
    createBrick2();
    console.log(bricks_level2);
  } else if (level === 2 && score === 210) {
    var congratulations = this.add.text(
      game.config.width / 2,
      game.config.height / 3,
      "Congratulations!!!\n You finish the game!",
      { font: "32px Arial", fill: "ffffff" }
    );
    congratulations.setOrigin(0.5);

    var restartButton = this.add.text(
      game.config.width / 2,
      game.config.height / 3 + 150,
      "Jogar Novamente!",
      {
        font: "32px Arial",
        fill: "ffffff",
        stroke: "ffffff",
        backgroundColor: "#8878C3",
      }
    );
    restartButton.setOrigin(0.5);

    restartButton.setInteractive(); // Habilita a interatividade do botão
    restartButton.on("pointerup", function () {
      window.location.href = "index.html";
    });

    this.physics.pause();
    this.input.mouse.disableContextMenu();
    this.input.keyboard.enabled = false;
  }
}
```

Figura 11: Update

3.1.5 BounceOffPaddle

Esta função é responsável por fazer a bola saltar quando esta colide com o paddle. Para além disso conforme o nível em que o jogador se encontra a velocidade do embate entre a bola e o paddle também aumenta, para que assim a dificuldade também aumente.

```
function bounceOffPaddle() {  
  if (level === 1) {  
    ball.setVelocityY(-300);  
    ball.setVelocityX(Phaser.Math.Between(-350, 350));  
  } else if (level === 2) {  
    ball.setVelocityY(-500);  
    ball.setVelocityX(Phaser.Math.Between(-550, 550));  
  } else if (level === 3) {  
    ball.setVelocityY(-700);  
    ball.setVelocityX(Phaser.Math.Between(-750, 750));  
  }  
}
```

Figura 12: BounceOffPaddle

3.1.6 CreateBricks

A função "CreateBricks" é responsável por criar um conjunto de blocos no jogo, para isso é utilizada uma matriz padrão dos blocos que devem ser criados.

Após a criação da matriz padrão, a função percorre cada elemento da matriz e verifica se o valor na matriz é igual a 1, caso se verifique, a função cria um bloco nas coordenadas indicadas através das variáveis "brickX" e "brickY". Por último os blocos são adicionados à física do jogo e a função "manage" é chamada.

```
function createBricks1() {  
  bonusCount = 0;  
  var brickOffset = {  
    top: 50,  
    left: 115,  
  };  
  var brickMatrix = [  
    [1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1],  
    [1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1],  
    [0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1],  
    [0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1],  
    [1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1],  
    [1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1],  
    [1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1],  
  ];  
  var rows = brickMatrix.length;  
  var cols = brickMatrix[0].length;  
  // Loop através da matriz de padrão de blocos  
  for (var i = 0; i < rows; i++) {  
    for (var j = 0; j < cols; j++) {  
      // Se o valor na matriz for 1 ou 2, cria um bloco  
      if (brickMatrix[i][j] == 1 || brickMatrix[i][j] == 2) {  
        var brickX = j * (50 + 7) + brickOffset.left;  
        var brickY = i * (20 + 7) + brickOffset.top;  
        var brickColor = Phaser.Utils.Array.GetRandom(color);  
        var brick = scene.physics.add.existing(  
          scene.add.rectangle(brickX, brickY, 50, 20, brickColor)  
        );  
        manage(brick);  
      }  
    }  
  }  
}
```

```
function createBricks2() {  
  bonusCount = 0;  
  var brickOffset = {  
    top: 50,  
    left: 115,  
  };  
  var brickMatrix = [  
    [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],  
    [0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0],  
    [0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0],  
    [0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0],  
    [1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1],  
    [1, 1, 0, 0, 0, 0, 0, 1, 1, 1],  
    [1, 1, 0, 0, 0, 0, 0, 1, 1, 1],  
    [1, 1, 0, 0, 0, 0, 0, 1, 1, 1],  
    [0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0],  
    [0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0],  
    [0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0],  
  ];  
  var rows = brickMatrix.length;  
  var cols = brickMatrix[0].length;  
  // Loop através da matriz de padrão de blocos  
  for (var i = 0; i < rows; i++) {  
    for (var j = 0; j < cols; j++) {  
      // Se o valor na matriz for 1 ou 2, cria um bloco  
      if (brickMatrix[i][j] == 1 || brickMatrix[i][j] == 2) {  
        var brickX = j * (50 + 7) + brickOffset.left;  
        var brickY = i * (20 + 7) + brickOffset.top;  
        var brickColor = Phaser.Utils.Array.GetRandom(color);  
        var brick = scene.physics.add.existing(  
          scene.add.rectangle(brickX, brickY, 50, 20, brickColor)  
        );  
        manage(brick);  
      }  
    }  
  }  
  ball.setVelocity(500, 500);  
}
```

Figura 13: CreateBricks

3.1.7 Manage e BallHitBrick

A função "manage" é chamada para cada bloco criado e define que o bloco é imóvel e adiciona um evento de colisão com a bola que chama a função "ballHitBrick", que é chamada quando a bola colide com um bloco. Essa função verifica a cor do bloco, caso a cor seja branca, a velocidade da bola é aumentada, após isso o bloco é destruído e o score incrementado, onde após a destruição do bloco é retornado um valor de 0 a 1 através do "Math.random" em que se o valor for inferior a 0.2 ele invoca a função createBonus.

```
function manage(brick) {
  brick.body.immovable = true;
  scene.physics.add.collider(ball, brick, function () {
    ballHitBrick(brick);
  });
}

//Função para colisão da bola com o bloco
function ballHitBrick(brick) {
  if (brick.fillColor == "0xffffff") {
    ball.setVelocity(450, 450);
  }
  brick.destroy();
  bricks_level--;
  bricks_level1--;
  sound.play();
  sound.volume = 0.005;
  score++;
  scoreText.setText("Score: " + score);
  callBonus = Math.random() < 0.2 ? createBonus(brick.x, brick.y) : null;

  if (level === 2) {
    ball.setVelocity(450, 450);
  } else if (level === 2 && brick.fillColor == "0xffffff") {
    ball.setVelocity(550, 550);
  } else if (level === 3 && brick.fillColor == "0xffffff") {
    ball.setVelocity(650, 650);
  }
}
```

Figura 14: Manage e BallHitBrick

3.1.8 Informations e hitLava

A função "informations" cria e exibe o texto do score e das vidas na tela. A função "hitLava" é chamada quando a bola toca a lava e diminui a quantidade de vidas, atualizando a exibição das vidas na tela.

```
function informations() {
  //adicionar o texto do score e vidas
  scoreText = scene.add.text(16, 16, "Score: " + score, {
    fontSize: "32px",
    fill: "#FFF",
  });
  livesText = scene.add.text(630, 16, "Lives: " + lives, {
    fontSize: "32px",
    fill: "#FFF",
  });
}

function hitLava() {
  lives--;
  livesText.setText("Lives: " + lives);
}
```

Figura 15: Informations e hitLava

3.1.9 Create Bonus

A função “createBonus” recebe dois parâmetros, x e y, que indicam a posição onde o bônus deve ser criado. A variável “maxBonusPerLevel” indica a quantidade máxima de bônus que podem ser criados por nível. A variável “bonusType” é definida aleatoriamente com base nos dois tipos de bônus existentes, positivo ou negativo.

Em seguida, o código verifica se o número de bônus criados no nível atual ainda não atingiu o seu limite máximo. Se não, a variável “bonus” é criada como uma nova “sprite”. A textura da mesma, é definida de acordo com o tipo de bônus e o tamanho é ajustado para 0,05. A “sprite” é movida verticalmente com uma velocidade de 150 pixels por segundo.

Em seguida, é adicionado um detetor de colisão entre a “sprite” do bônus e a “sprite” do “paddle”. Quando há colisão, a “sprite” do bônus é desativada e suas propriedades físicas são removidas. Se o tipo de bônus for positivo a função “positiveBonus” será executada, caso contrário a função “negativeBonus” será executada.

```
function createBonus(x, y) {
  var maxBonusPerLevel = 5;
  bonusType = Math.random() < 0.5 ? "positive" : "negative";

  if (bonusCount < maxBonusPerLevel) {
    bonus = scene.physics.add.sprite(x, y, "bonus");
    if (bonusType === "positive") {
      bonus.setTexture("bonusPositive");
    } else if (bonusType === "negative") {
      bonus.setTexture("bonusNegative");
    }
    bonus.setScale(0.05);
    bonus.setVelocityY(150);

    // 0 seconds ago + 1/2
    scene.physics.add.collider(paddle, bonus, function (paddle, bonus) {
      bonus.disableBody(true, true);

      if (bonusType === "positive") {
        positiveBonus();
      } else if (bonusType === "negative") {
        negativeBonus();
      }
    });
    bonusCount++;
  }
}
```

Figura 16: Create Bonus

3.1.10 Positive Bonus

A função “positiveBonus” é chamada quando o jogador apanha um bônus positivo no jogo. A variável “bonusGoodActive”, quando verdadeira, indica que um bônus positivo está ativo no momento. A função “Phaser.Math.RND.pick” escolhe aleatoriamente entre dois possíveis bônus positivos: aumentar o tamanho do “paddle” ou receber uma vida extra. Se o número aleatório escolhido for 0, o tamanho do “paddle” é aumentado para 1,5 vezes o tamanho original. A função “scene.time.delayedCall” é usada para agendar uma chamada de função que será executada durante 10 segundos, após os 10 segundos a função retorna o tamanho do “paddle” ao normal e define a variável “bonusGoodActive” como falsa, indicando que o bônus positivo deixou de estar ativo. Se o número aleatório escolhido for 1, uma vida extra é adicionada e o texto de exibição das vidas é atualizado.

```
function positiveBonus() {
    // Verifica se o bônus positivo está ativo
    if (bonusGoodActive === true) {
        // Se o bônus positivo já está ativo, não faz nada
        return;
    }
    // Escolhe aleatoriamente entre dois bônus positivos
    let pick = Phaser.Math.RND.pick(0, 1);
    if (pick === 0) {
        // Aumenta o tamanho do paddle
        paddle.scale.x *= 1.5;
        paddle.scale.y *= 1.5;
    } else {
        // Adiciona uma vida extra
        lives += 1;
        updateLives();
    }
    // Agendar a chamada de função para 10 segundos
    scene.time.delayedCall(10, function() {
        // Retorna o tamanho do paddle ao normal
        paddle.scale.x = 1;
        paddle.scale.y = 1;
    });
    // Define a variável bonusGoodActive como falsa
    bonusGoodActive = false;
}
```

Figura 17: Positive Bonus

3.1.11 Negative Bonus

A função “negativeBonus” é chamada quando o jogador pega um bônus negativo no jogo. Inicialmente, verifica se já existe uma bola extra em jogo, se não houver, a função define a variável “bonusBadActive” como verdadeira, indicando que um bônus negativo está ativo. Em seguida, define a variável “extraBallActive” como verdadeira, indicando que uma bola extra foi adicionada ao jogo. O “extraBall” é configurado para colidir com o “paddle” e com a lava, e é removido do jogo após 5 segundos usando a função “setTimeout”. Se já houver uma bola extra em jogo, o tamanho do “paddle” é reduzido para metade. A função “scene.time.delayedCall” é usada para agendar uma chamada de função que será executada durante 10 segundos, onde após isso retorna o tamanho do “paddle” ao normal.

```
function negativeBonus() {
    // Verifica se o bônus negativo está ativo
    if (bonusBadActive === true) {
        // Se o bônus negativo já está ativo, não faz nada
        return;
    }
    // Define a variável bonusBadActive como verdadeira
    bonusBadActive = true;
    // Define a variável extraBallActive como verdadeira
    extraBallActive = true;
    // Cria a bola extra
    let extraBall = scene.physics.add.sprite(100, 100, 'ball');
    // Configura a bola extra para colidir com o paddle e a lava
    extraBall.setCollideEvents(paddle, true);
    extraBall.setCollideEvents(lava, true);
    // Define o tempo de vida da bola extra
    setTimeout(function() {
        // Remove a bola extra do jogo
        scene.physics.remove(extraBall);
    }, 5000);
    // Agendar a chamada de função para 10 segundos
    scene.time.delayedCall(10, function() {
        // Retorna o tamanho do paddle ao normal
        paddle.scale.x = 1;
        paddle.scale.y = 1;
    });
}
```

Figura 18: Negative Bonus

3.1.12 HandleInput e SetupInput

A função “handleInput” é chamada quando se pressiona uma tecla ou se mexe no rato. Quando um destes acontece, a função verifica qual foi, se foi a seta esquerda, a seta direita ou se mexeu o rato. Se foi uma das setas, atribui a velocidade, se foi o movimento do rato, atualiza a posição do “paddle” para o local onde o cursor se encontra.

```
function handleInput(event) {
  if (event.type === "keydown") {
    if (event.code === "ArrowLeft") {
      paddle.setVelocityX(-300);
    } else if (event.code === "ArrowRight") {
      paddle.setVelocityX(300);
    }
  } else if (event.type === "mousemove") {
    scene.input.on("pointermove", function (pointer) {
      paddle.setPosition(pointer.x, paddle.y);
    });
  }

  if (event.type === "keyup") {
    if (event.code === "ArrowLeft" && paddle.body.velocity.x < 0) {
      paddle.setVelocityX(0);
    } else if (event.code === "ArrowRight" && paddle.body.velocity.x > 0) {
      paddle.setVelocityX(0);
    }
  }
}

// cria os ouvintes de eventos para teclado e mouse
function setupInput() {
  window.addEventListener("keydown", handleInput);
  window.addEventListener("mousemove", handleInput);
}
```

Figura 19: HandleInput e SetupInput

4 Conclusão

Em suma, a realização do exercício proposto foi algo muito positivo para o desenvolvimento das nossas aptidões no que toca à biblioteca Phaser3.

Com a realização deste projeto, conseguimos por em prática muitas das coisas que fomos aprendendo no decorrer das aulas, porém também levou a que procurássemos saber mais acerca do quão longe a biblioteca Phaser3 nos podia levar, e para que pudessemos resolver as diferentes tarefas que o jogo necessitava.

Para finalizar, o jogo poderia estar um pouco melhor conseguido, no entanto penso que cumpre com a maioria dos requisitos propostos pelo professor. Futuramente gostávamos de conseguir resolver o bug existente no jogo, e implementar novos recursos para que seja possível proporcionar ao utilizador uma melhor usabilidade do mesmo.

5 Links

GitHub: Repositório do Jogo.

Netlify: Link do Jogo Online.