ImpactJS e Phaser

Aluno: André Correia Zarzur e Henrique Oliveira de Souza

Professor: Johnata Souza Santicioli

Impact

Introdução

Framework de jogos 2D criado por Dominic Szablewski (2010)

Elemento `<canvas>`

Código aberto (MIT) desde 2014

Freeman: fácil aprendizado, comunidade ativa e Ejecta

Impact

Conceitos-chave

Módulos

Classes

Editor de níveis (Weltmeister)

Sprites

Colisão

Animações

Impact

Desenvolvimento

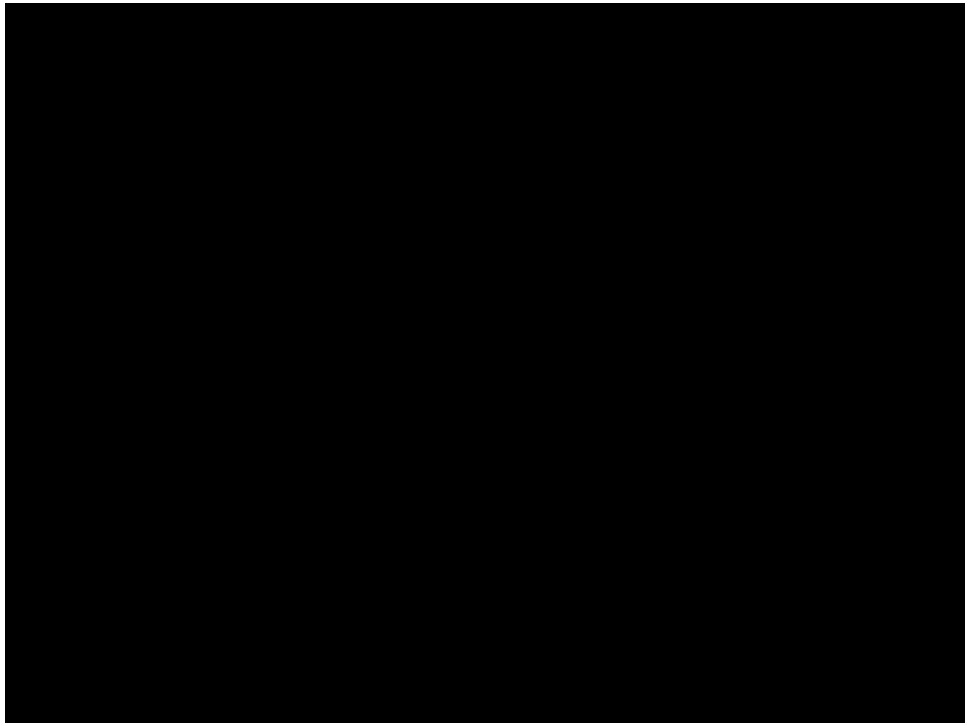
Configuração do ambiente

1. Docker (PHP 5 e Apache), VS Code (Live Server)
2. Download do Impact Engine `https://github.com/phoboslab/impact`.
    2.1. lib > game > entities
    2.2. media
    2.3. tools

```
// Entidade Paddle
// ...
ig.module(
    'game.entities.paddle'
  )
.requires(
    'impact.entity'
)
.defines(function(){
  EntityPaddle  = ig.Entity.extend({
        size: { x : 64, y: 128 },
        collides: ig.Entity.COLLIDES.FIXED,

        animSheet: new ig.AnimationSheet  ('media/paddle-red.png' , 64, 128),

        init: function(x, y, settings) {
            this.parent(x, y, settings);

            this.addAnim('idle', 1, [0]);
        }
  });
});
```

Impact

Exemplo



Phaser

Introdução

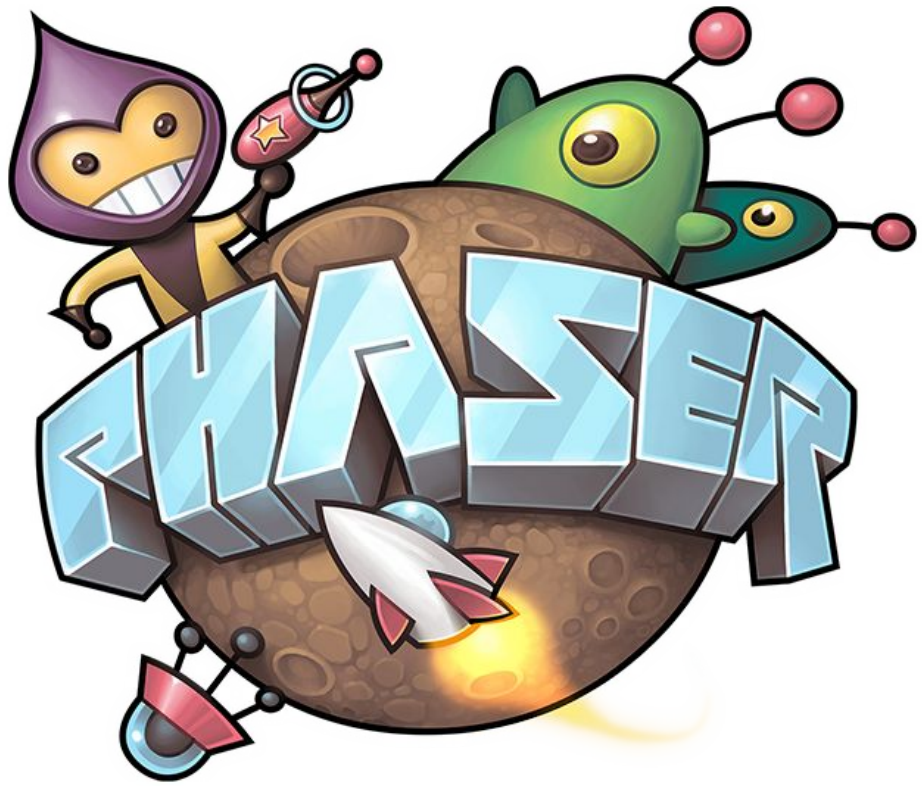
Framework de jogos 2D criado pela Photon Storm (2013)

Elemento `<canvas>` e WebGL

Código aberto

Fácil aprendizado, um dos frameworks para jogos mais populares do GitHub

Desenvolvimento pode ser feito em JavaScript ou TypeScript



Phaser

Conceitos-chave

Sprites

Colisão

Animações

Config (objeto para configurar o jogo, por exemplo a física)

Preload, Create, Update (funções)

Phaser

Desenvolvimento

Configuração do ambiente

1. Docker (Apache), VS Code (Live Server)
2. `<script src="//cdn.jsdelivr.net/npm/phaser@3.55.2/dist/phaser.js"></script>`

```
// Configuração
    var config = {
        type: Phaser.AUTO,
        width: 800,
        height: 600,
        physics: {
            default: 'arcade',
            arcade: {
                gravity: { y: 300 },
                debug: false
                }
    },
        scene: {
            preload: preload,
            create: create,
            update: update
        }
    };

//  Arcade é o tipo de física adicionada ao
jogo, nesse caso essa é a melhor pra esse
tipo de jogo por ser simples e não precisar
de estabilidade
```

```
// Declaração de variáveis / criação do objeto Game
    var player;
    var stars;
    var bombs;
    var platforms;
    var cursors;
    var score = 0;
    var gameOver = false;
    var scoreText;

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

```
// Carregando elementos
// …
function preload ()
{
    this.load.image('sky', 'assets/sky.png');
    this.load.image('ground', 'assets/platform.png');
    this.load.image('star', 'assets/star.png');
    this.load.image('bomb', 'assets/bomb.png');
    this.load.spritesheet('dude', 'assets/dude.png', { frameWidth: 32, frameHeight: 48 });
}
```

```
// Adicionando os elementos
// …
function create ()
{
    this.add.image(400, 300, 'sky');

    //  Adiciona a colisão do tipo estática as plataformas
    platforms = this.physics.add.staticGroup();

    platforms.create(400, 568, 'ground').setScale(2).refreshBody();
    platforms.create(600, 400, 'ground');
    platforms.create(50, 250, 'ground');
    platforms.create(750, 220, 'ground');

    //  Adiciona o jogador na posição indicada
    player = this.physics.add.sprite(100, 450, 'dude');

    //  Faz o player quicar quando ele cai no chão e evita que ele saia do display estabelecido
(800x600)
    player.setBounce(0.2);
    player.setCollideWorldBounds(true);
}
```

```
// Adicionando os elementos
// …
function create ()
{
    //  Adiciona as animações ao player
    this.anims.create({
        key: 'left',
        frames: this.anims.generateFrameNumbers('dude', { start: 0, end: 3 }),
        frameRate: 10,
        repeat: -1
    });

    this.anims.create({
        key: 'turn',
        frames: [ { key: 'dude', frame: 4 } ],
        frameRate: 20
    });

    this.anims.create({
        key: 'right',
        frames: this.anims.generateFrameNumbers('dude', { start: 5, end: 8 }),
        frameRate: 10,
        repeat: -1
    });
}
```

```
// Adicionando os elementos
// …
function create ()
{
    cursors = this.input.keyboard.createCursorKeys(); //  Recebe os inputs do teclado do jogador

    stars = this.physics.add.group({ //  Adiciona as 12 estrelas para coletar
        key: 'star',
        repeat: 11,
        setXY: { x: 12, y: 0, stepX: 70 }
    });

    stars.children.iterate(function (child) {
        child.setBounceY(Phaser.Math.FloatBetween(0.4, 0.8)); //  Cria um valor aleatório para a estrela quicar
    });

    bombs = this.physics.add.group();

    scoreText = this.add.text(16, 16, 'score: 0', { fontSize: '32px', fill: '#000' }); //  Adiciona a pontuação

    this.physics.add.collider(player, platforms); //  Cria a colisão do player/estrela/bomba com a plataforma
    this.physics.add.collider(stars, platforms);
    this.physics.add.collider(bombs, platforms);

    this.physics.add.overlap(player, stars, collectStar, null, this); //  Checa se o jogador entrou em contato com
                                                                       //  uma estrela
    this.physics.add.collider(player, bombs, hitBomb, null, this);    //  O mesmo, mas com a bomba
}
```

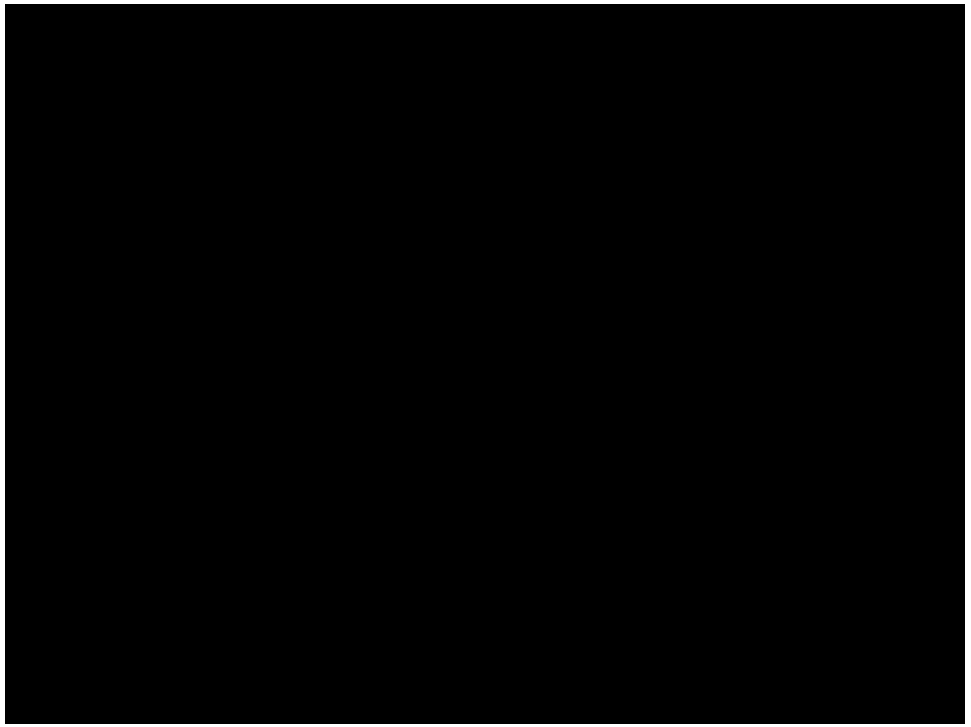
```
// Adicionar movimento ao player e o game over
// …
function update ()
{
    if (gameOver) {
        return;
    }

    if (cursors.left.isDown) {
        player.setVelocityX(-160);
        player.anims.play('left', true);
    } else if (cursors.right.isDown) {
        player.setVelocityX(160);
        player.anims.play('right', true);
    } else {
        player.setVelocityX(0);
        player.anims.play('turn');
    }

    if (cursors.up.isDown && player.body.touching.down) {
        player.setVelocityY(-330);
    }
}
```

```
// Adicionar a função para coletar as estrelas
// …
function collectStar (player, star)
{
    star.disableBody(true, true); //  Faz a estrela desaparecer

    score += 10; //  Atualiza a pontuação
    scoreText.setText('Score: ' + score);

    if (stars.countActive(true) === 0) { //  Caso todas as estrelas tenham sido coletadas
        stars.children.iterate(function (child) { //  Cria novo grupo de estrelas
            child.enableBody(true, child.x, 0, true, true);
        });

        //  Calculo para ver onde a bomba irá aparecer
        var x = (player.x < 400) ? Phaser.Math.Between(400, 800) : Phaser.Math.Between(0, 400);

        var bomb = bombs.create(x, 16, 'bomb'); //  Cria a bomba
        bomb.setBounce(1);
        bomb.setCollideWorldBounds(true);
        bomb.setVelocity(Phaser.Math.Between(-200, 200), 20);
        bomb.allowGravity = false;
    }
}
```

```
// Adicionar a função para quando o player encosta na bomba
// …
function hitBomb ()
{
    this.physics.pause(); //  Para toda a física do jogo

    player.setTint(0xff0000); //  Muda cor do player para vermelho

    player.anims.play('turn'); //  Coloca o player no sprite neutro (de frente)

    gameOver = true; //  Aciona o game over
}
```

Phaser

Exemplo



Impact e Phaser

Referências bibliográficas

FREEMAN, J. Introducing HTML5 Game Development. O'Reilly Media, Inc. 2012. 1a edição. Califórnia.

SZABLEWSKI, D. Documentation - Impact. Disponível em: <https://impactjs.com/documentation>. Acesso em: 29 out. 2021.

DAVEY, R. Making your first Phaser 3 game. Disponível em: <https://phaser.io/tutorials/making-your-first-phaser-3-game-portuguese>. Acesso em: 24 out. 2021.