ImpactJs

E

Phaser

Codificação de games

Introdução

Tópicos - IMPACT

01

Overview

Descrição da tecnologia

02

Licenças

Preços e níveis de licença necessários

03

Casos de Uso

Plataformas criadas utilizando a tecnologia

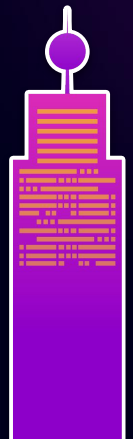
04

Exemplos

Exemplos práticos de aplicação da tecnologia

01

Overview



O que é?

Criador

Utilização

Ambiente

Popularização

02

Licenças

De serviço pago à OpenSource

2002 - 2014

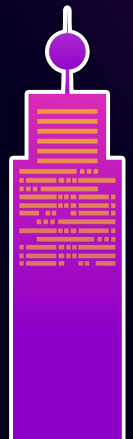
Desde seu lançamento até o ano de 2014 as licenças necessárias para utilizar os softwares específicos para a utilização da framework custavam em média $99

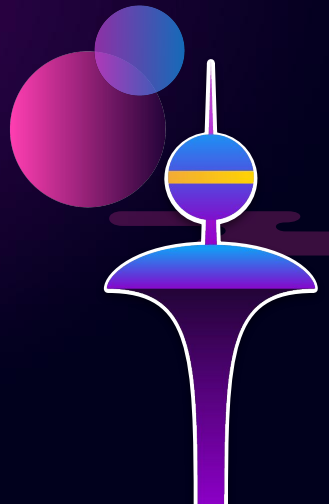
2014 - Atual

A partir de 2014, a utilização de softwares para fazer a codificação em ImpactJs passou a ser OpenSource, de acordo com a licença MIT

03

Casos de Uso





Jogos 2D em Javascript?

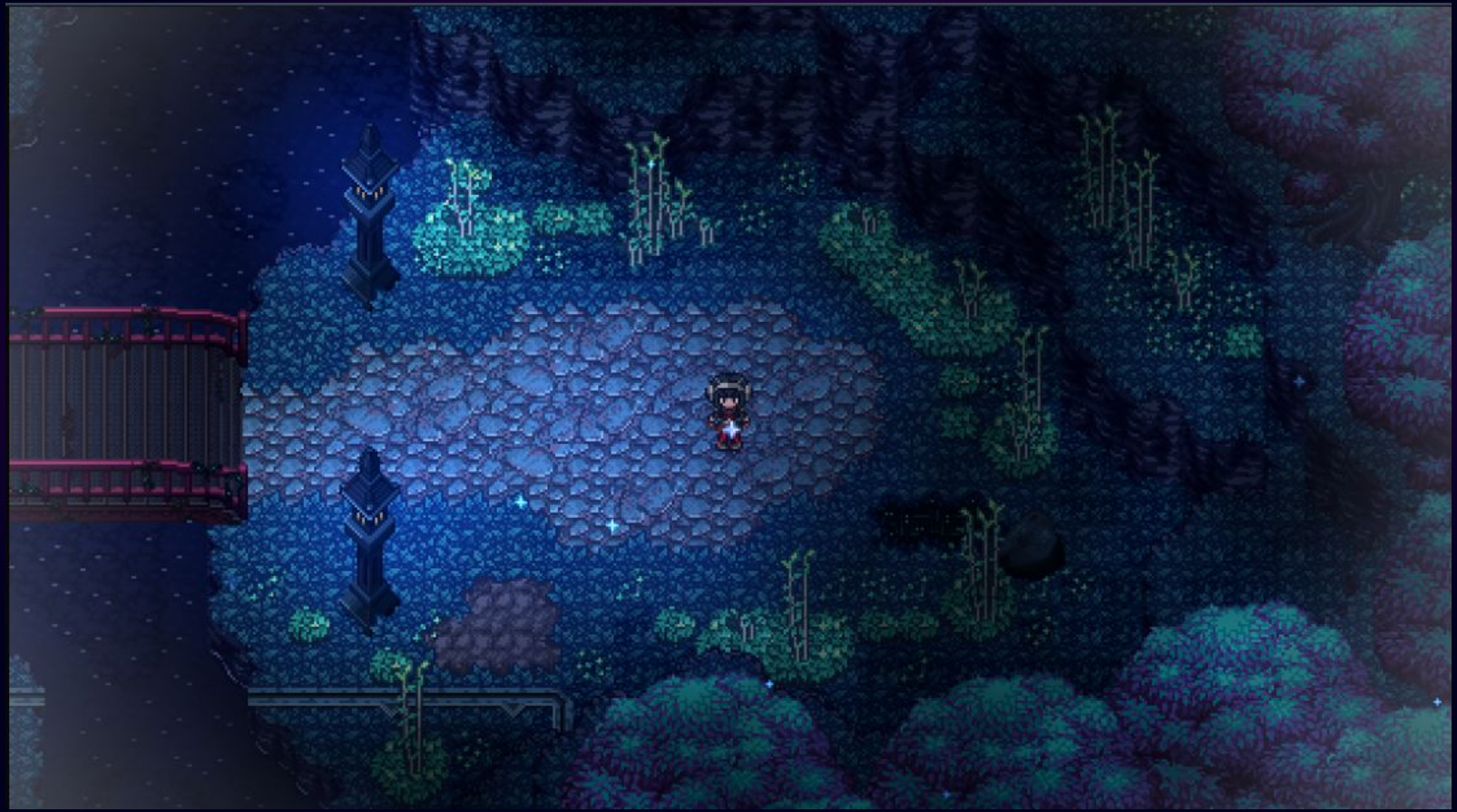
A engine Impact pode ser utilizada para desenvolver os mais diversos jogos 2D, desde jogos de plataforma à RPGs complexos, ou mesmo shooters top down.

Preco - Desvie enquanto cai

W.W.W. Bandits - Um cowboy que desvia de flechas, tiros e...

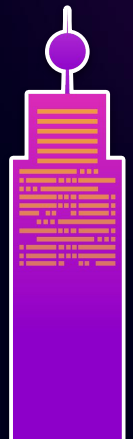
УПРАВЛЕНИЕ - Inspirado em lendas mitológicas

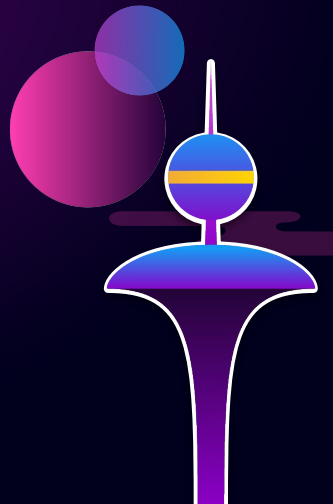
CrossCode - Um RPG completo

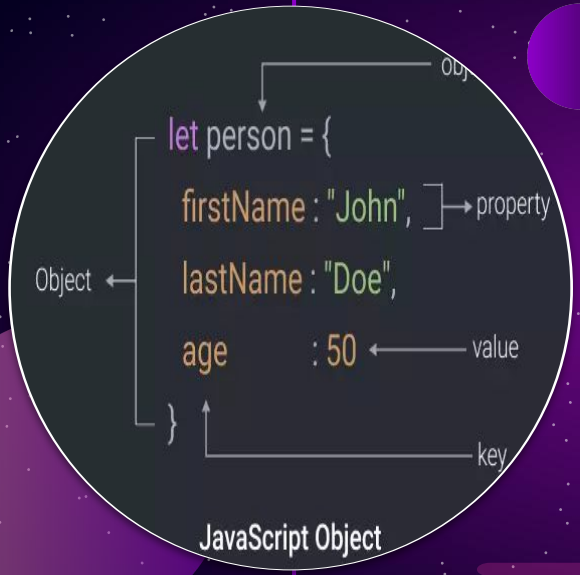


04

Exemplos







Impact's Classes

Em JavaScript, não há uma real estrutura de classe tradicional como você tem em outras linguagens OOP. Para resolver esse problema, o Impact possui um objeto pseudo-classe, que é a base de todas as classes utilizadas ao criar um jogo.

Classes mais utilizadas

- ig.Animation
- ig.Entity
- ig.Sound
- ig.Game
- ig.Input

```
ig.Animation = ig.Class.extend({
    sheet: null,
    timer: null,

    sequence: [],
    flip: {x: false, y: false},
    pivot: {x: 0, y: 0},

    frame: 0,
    tile: 0,
    loopCount: 0,
    alpha: 1,
    angle: 0,


    init: function( sheet, frameTime, sequence, stop ) {
        this.sheet = sheet;
        this.pivot = {x: sheet.width/2, y: sheet.height/2 };
        this.timer = new ig.Timer();

        this.frameTime = frameTime;
        this.sequence = sequence;
        this.stop = !!stop;
        this.tile = this.sequence[0];
    },
```

ig.Animation

Essa é uma classe utilizada para realizar a configuração da animação de uma sequência de sprints de um personagem ou background, auxiliada pela classe ig.AnimationSheet que armazena os parâmetros utilizados em cada frame de uma sprint.

ig.Entity

A classe ig.Entity é utilizada para armazenar todos os parâmetros relacionados aos personagens, bem como os parâmetros de game objects que podem ser interagidos durante o jogo. Dentre entre esses parâmetros estão: Físicas aplicadas aos objetos, colisores, valores de aceleração, valor de repulsão, entre outros.

```
ig.Entity = ig.Class.extend({
    id: 0, settings: {}, size: {x: 16, y:16}, offset: {x: 0, y: 0},

    pos: {x: 0, y:0}, last: {x: 0, y:0}, vel: {x: 0, y: 0}, accel: {x: 0, y: 0},
    friction: {x: 0, y: 0}, maxVel: {x: 100, y: 100}, zIndex: 0, gravityFactor: 1,
    standing: false, bounciness: 0, minBounceVelocity: 40,

    anims: {}, animSheet: null, currentAnim: null, health: 10,

    type: 0, // TYPE.NONE
    checkAgainst: 0, // TYPE.NONE
    collides: 0, // COLLIDES.NEVER

    _killed: false,

    slopeStanding: {min: (44).toRad(), max: (136).toRad() },

    init: function( x, y, settings ) {
        this.id = ++ig.Entity._lastId;
        this.pos.x = this.last.x = x;
        this.pos.y = this.last.y = y;

        ig.merge( this, settings );
    },
```

```
ig.SoundManager = ig.Class.extend({
    clips: {},
    volume: 1,
    format: null,

    init: function() {
        // Quick sanity check if the Browser supports the Audio tag
        if( !ig.Sound.enabled || !window.Audio ) {
            ig.Sound.enabled = false;
            return;
        }

        // Probe sound formats and determine the file extension to load
        var probe = new Audio();
        for( var i = 0; i < ig.Sound.use.length; i++ ) {
            var format = ig.Sound.use[i];
            if( probe.canPlayType(format.mime) ) {
                this.format = format;
                break;
            }
        }

        // No compatible format found? -> Disable sound
        if( !this.format ) {
            ig.Sound.enabled = false;
        }
```

ig.Sound

ig.Sound é uma classe utilizada para fazer a configuração de sons durante uma gameplay, desde quais os áudios que serão reproduzidos, até adaptação de volume e efeitos aplicados nos mesmos. Essa classe em específico apenas irá desempenhar suas funções se o navegador/plataforma utilizada tiver algum tipo de suporte a áudio.

ig.Game

Sendo uma das principais classes em um jogo, a ig.Game desempenha o papel de controle de variáveis, inicializando algumas funções e sendo a controladora da “HUB” do jogo. Essa é uma classe muito versátil que pode conter informações de diversas outras classes.

```
ig.Game = ig.Class.extend({

    clearColor: '#000000', gravity: 0, screen: {x: 0, y: 0}, _rscreen: {x: 0, y: 0}, entities: [],
    namedEntities: {}, collisionMap: ig.CollisionMap.staticNoCollision,
    backgroundMaps: [], backgroundAnims: {}, autoSort: false, sortBy: null, cellSize: 64,
    _deferredKill: [], _levelToLoad: null, _doSortEntities: false,

    staticInstantiate: function() {
        this.sortBy = this.sortBy || ig.Game.SORT.Z_INDEX;
        ig.game = this;
        return null;
    },

    loadLevel: function( data ) {
        this.screen = {x: 0, y: 0};

        // Entities
        this.entities = [];
        this.namedEntities = {};
        for( var i = 0; i < data.entities.length; i++ ) {
            var ent = data.entities[i];
            this.spawnEntity( ent.type, ent.x, ent.y, ent.settings );
        }
        this.sortEntities();

        // Map Layer
        this.collisionMap = ig.CollisionMap.staticNoCollision;
```

```
ig.Input = ig.Class.extend({
    bindings: {}, actions: {}, presses: {}, locks: {}, delayedKeyup: {},

    isUsingMouse: false, isUsingKeyboard: false, isUsingAccelerometer: false,
    mouse: {x: 0, y: 0}, accel: {x: 0, y: 0, z: 0},

    initMouse: function() {
        if( this.isUsingMouse ) { return; }
        this.isUsingMouse = true;
        var mouseWheelBound = this.mousewheel.bind(this);
        ig.system.canvas.addEventListener('mousewheel', mouseWheelBound, false );
        ig.system.canvas.addEventListener('DOMMouseScroll', mouseWheelBound, false );

        ig.system.canvas.addEventListener('contextmenu', this.contextmenu.bind(this), false );
        ig.system.canvas.addEventListener('mousedown', this.keydown.bind(this), false );
        ig.system.canvas.addEventListener('mouseup', this.keyup.bind(this), false );
        ig.system.canvas.addEventListener('mousemove', this.mousemove.bind(this), false );

        if( ig.ua.touchDevice ) {
            // Standard
            ig.system.canvas.addEventListener('touchstart', this.keydown.bind(this), false );
            ig.system.canvas.addEventListener('touchend', this.keyup.bind(this), false );
            ig.system.canvas.addEventListener('touchmove', this.mousemove.bind(this), false );

            // MS
            ig.system.canvas.addEventListener('MSPointerDown', this.keydown.bind(this), false );
            ig.system.canvas.addEventListener('MSPointerUp', this.keyup.bind(this), false );
            ig.system.canvas.addEventListener('MSPointerMove', this.mousemove.bind(this), false );
            ig.system.canvas.style.msTouchAction = 'none';
        }
    },
```

ig.Input

Uma classe de extrema importância dentro de qualquer jogo que utilize Impact é a ig.Input, pois ela é responsável por toda a configuração de ações realizadas pelo usuário (ou jogador no caso), como por exemplo a captura de teclas pressionadas no teclado ou então botões do mouse.

Tópicos - 

01

Overview

Descrição da tecnologia

02

Licenças

Preços e níveis de licença necessários

03

Casos de Uso

Plataformas criadas utilizando a tecnologia

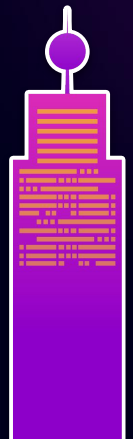
04

Exemplos

Exemplos práticos de aplicação da tecnologia

01

Overview



O que é?

Criador

Utilização

Ambiente

Popularização

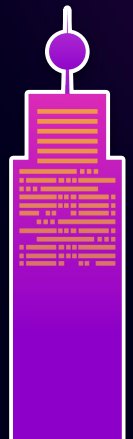
02

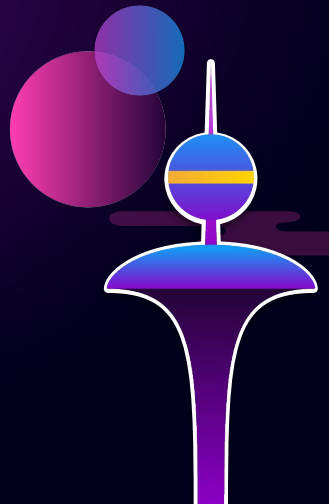
Licenças

Quanto custa para utilizar o Phaser?🚬

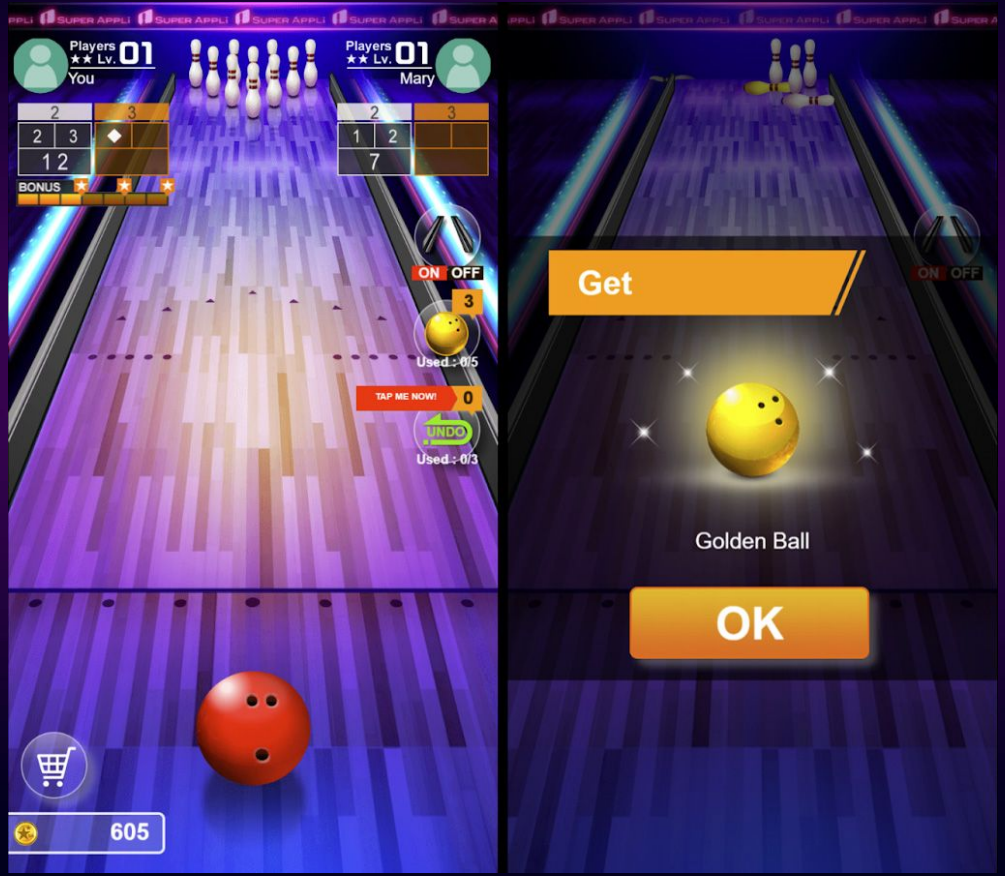
03

Casos de Uso

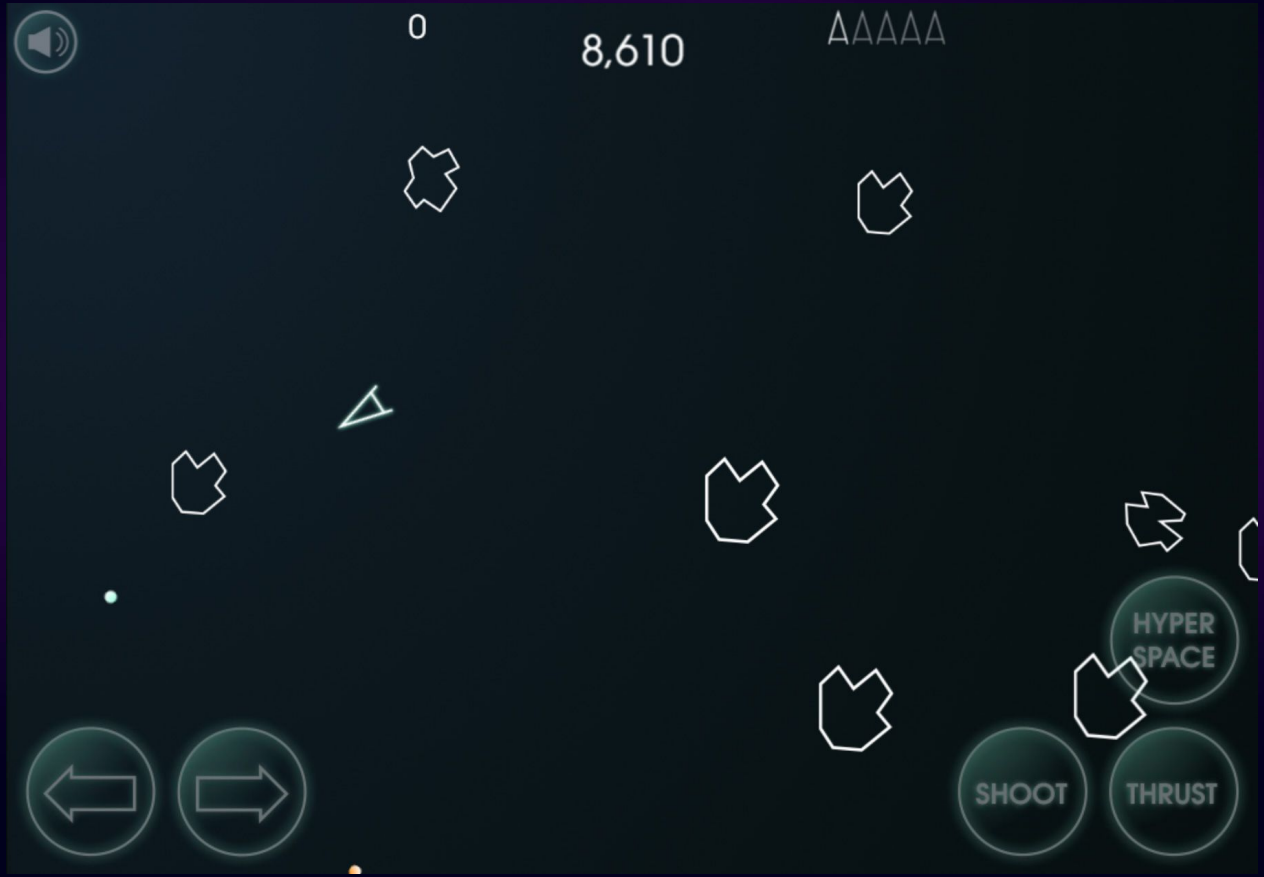




The Bowling Club



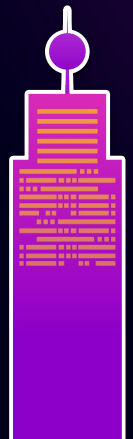
Atari Asteroids

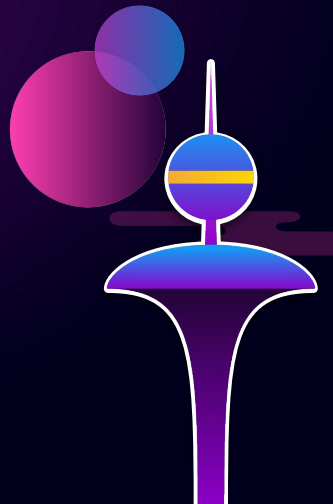


8-Ball Billiards

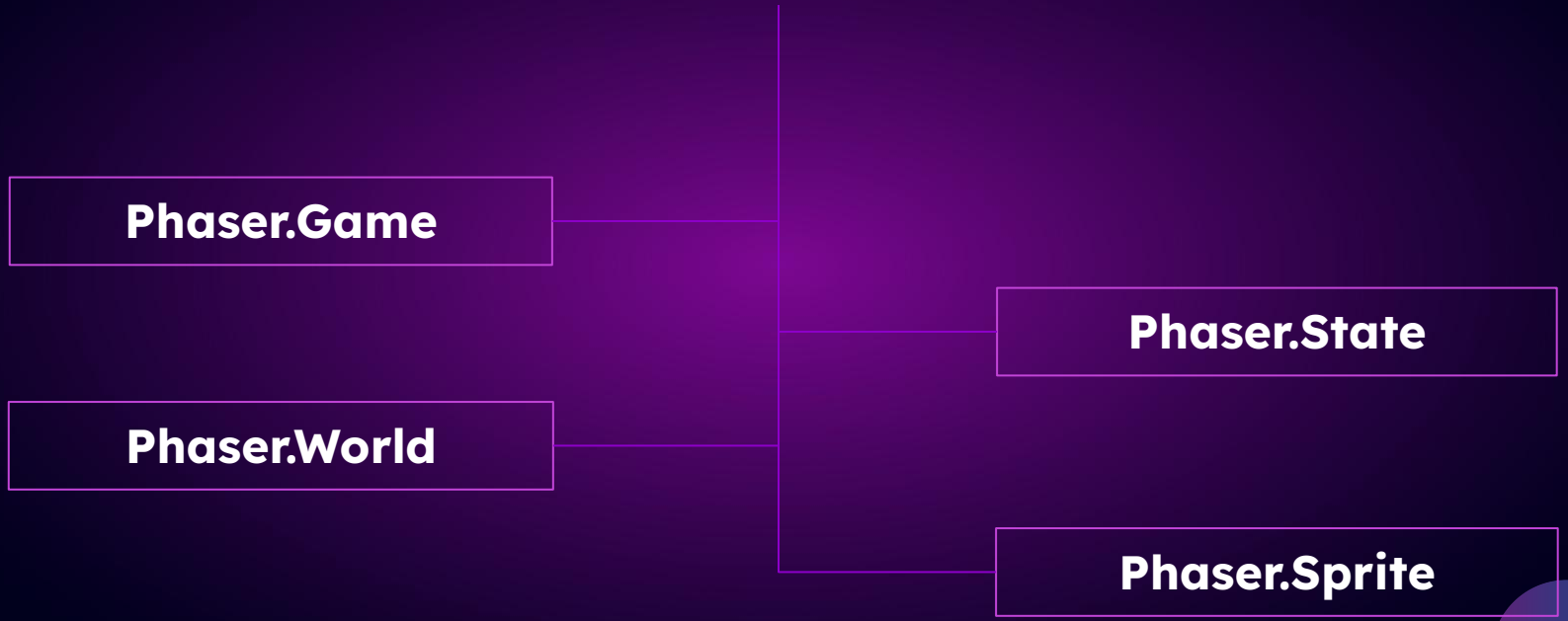
04

Exemplos





Classes mais utilizadas



```
<!doctype html>
<html lang="pt-br">
    <head>
        <meta charset="UTF-8" />
        <title>hello phaser!</title>
        <script src="../js/phaser.min.js"></script>
        <script type="text/javascript">
        var game = new Phaser.Game(800, 600, Phaser.AUTO, '',
                                   { preload: preload, create: create });

        function preload () {
            game.load.image('logo', '../images/phaser.png');
        }

        function create () {
            var logo = game.add.sprite(game.world.centerX, game.world.cente
            logo.anchor.setTo(0.5, 0.5);
        }
        </script>
    </head>
    <body>
    </body>
</html>
```

Phaser.Game

A classe Phaser.Game representa o controlador principal do jogo. Ela é responsável por tratar o processo de boot, analisar arquivos de configuração, criar o renderer e configurar todos os sistemas do Phaser, como física, entrada e som.

Outra característica importante é que o código cria o objeto game e o disponibiliza globalmente para todo o código.

```
Phaser.State.prototype = {

    init: function () {
    },

    preload: function () {
    },

    loadRender: function () {
    },

    create: function () {
    },

    update: function () {
    },

    pauseUpdate: function () {
    },

    shutdown: function () {
    }

};

Phaser.State.prototype.constructor = Phaser.State;
```

Phaser.State

A classe Phaser.State representa o estado do jogo. No caso do exemplo são tratados dois estados:

preload: tratado na função preload()
create: tratado na função create()

```
Phaser.World = function (game) {

    Phaser.Group.call(this, game, null, '__world', false);


    this.bounds = new Phaser.Rectangle(0, 0, game.width, game.height);

    this.camera = null;

    this._definedSize = false;

    this._width = game.width;

    this._height = game.height;

    this.game.state.onStateChange.add(this.stateChange, this);
};
```

Phaser.World

O world, classe Phaser.World, representa um local abstrato, no qual estão todos os objetos do jogo. As câmeras permitem "olhar" para o mundo, tornando os seus objetos visíveis.

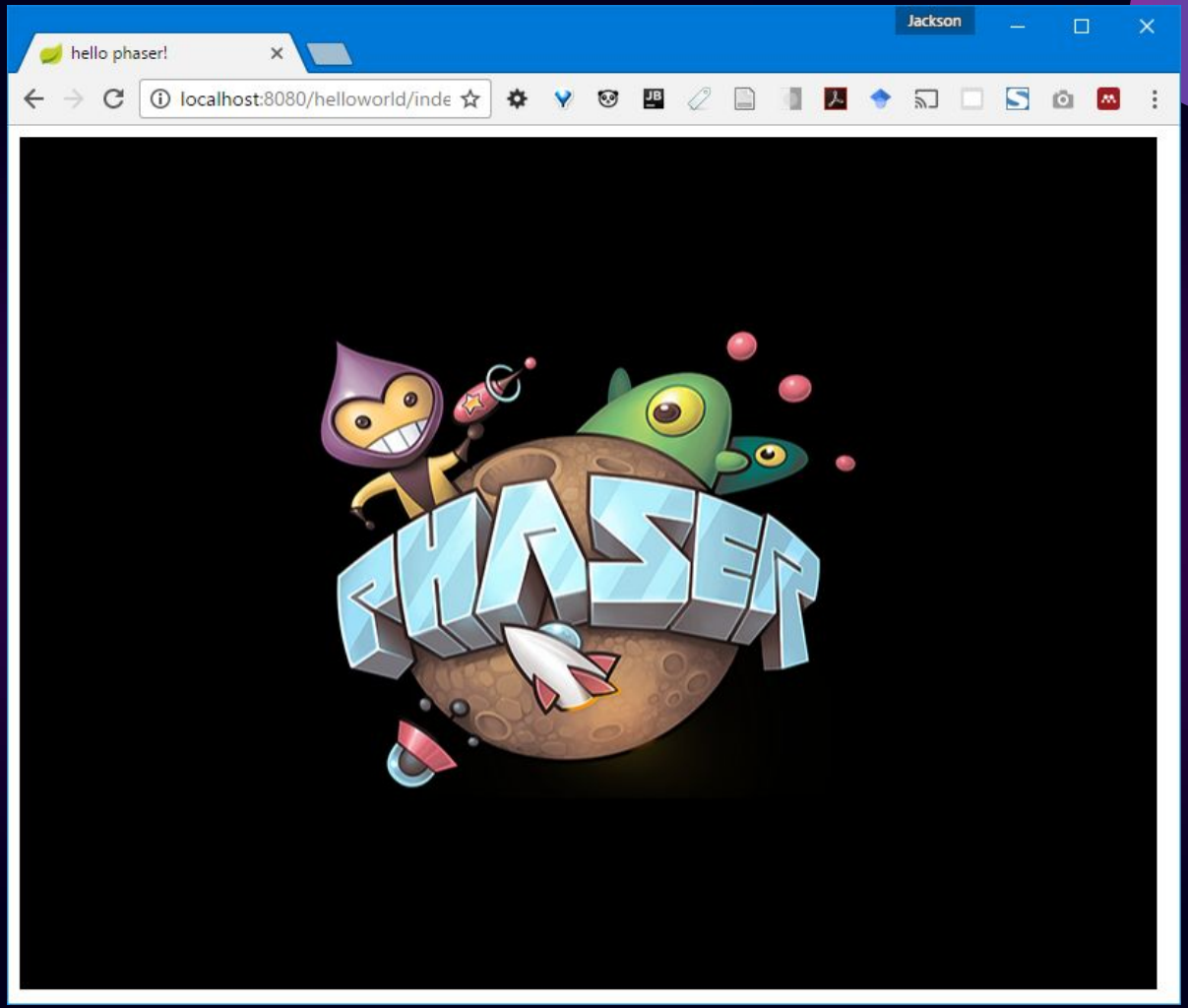
Phaser.Sprite

```
<!doctype html>
<html lang="pt-br">
    <head>
        <meta charset="UTF-8" />
        <title>hello phaser!</title>
        <script src="../js/phaser.min.js"></script>
        <script type="text/javascript">
        var game = new Phaser.Game(800, 600, Phaser.AUTO, '',
                                   { preload: preload, create: create });

        function preload () {
            game.load.image('logo', '../images/phaser.png');
        }

        function create () {
            var logo = game.add.sprite(game.world.centerX, game.world.cente
            logo.anchor.setTo(0.5, 0.5);
        }
        </script>
    </head>
    <body>
    </body>
</html>
```

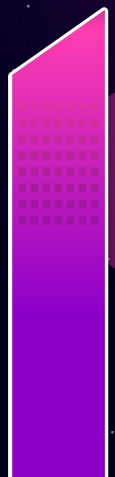
O sprite, classe Phaser.Sprite, representa uma parte vital do jogo, praticamente todo objeto visual. O sprite mais básico consiste de uma posição, coordenada (x,y), e uma textura que é renderizada no canvas. Além disso, contém propriedades para lidar, por exemplo, com física (Sprite.body), tratamento de entrada (Sprite.input), eventos (Sprite.events) e animação (Sprite.animations).


hello phaser!
Jackson
localhost:8080/helloworld/inde
PHASER

É hora da ação..

http://dontpad.com/impact_phaser

Comparação das frameworks



ImpactJs X Phaser

		ImpactJS	Phaser
01	Melhor ranking de framework para desenvolvimento de jogos em HTML5 (1° lugar)		✓
02	Maior facilidade de uso (indicado para iniciantes)		✓
03	Melhor documentação disponibilizada		✓
04	Melhor performance de execução	✓	

ImpactJs X Phaser

		ImpactJS	Phaser
01	Maior facilidade de implantação		✓
02	Melhor configuração de detalhes como impacto e movimentação de sprites	✓	
03	Mais recomendado para aplicações profissionais	✓	
04	Maior quantidade de plugins	✓	

Conclusão

Bibliografia

- https://phaser.io/tutorials/getting-started-phaser3
- http://www.photonstorm.com/contact
- https://github.com/photonstorm/phaser
- https://jacksongomesbr.gitbooks.io/computacao-grafica/content/cg-interativa/comecando-com-phaserio.html
- https://impactjs.com
- https://resources.oreilly.com/examples/
- https://www.treinaweb.com.br/blog/javascript-game-engines
- https://www.slant.co/versus/1061/1966/~impactjs_vs_phaser
- https://www.oreilly.com/library/view/building-html5-games/9781449331207/

Obrigado!