PhaserJS Game Engine Cheat sheet

Why should we take a look at PhaserJS, a javascript game engine when it comes to programming learning?

Well, grabbing a popular HMTL5 Game engine like PhaserJS could make you have a better understanding the hood under some well-known online learning platform like `Prodigy` and `Scratch`.

Moreover, web game is easy to get started and share with your friends, if you have great interests in game development and would like delve into this field, PhaserJS is a good starting point.

Actually, all the mini games in this website are made of PhaserJS game engine.

Phaser Game Config

Code	Comments
export const BaseConfig = {	
type: Phaser.AUTO,	
width: 600,	
height: 237,	
parent: 'phaser-game-box',	**// div id in webpage**
banner: false,	
};	

Game Scene Skeleton

Code
class TemplateGame extends Phaser.Scene {
constructor(){

Platformer Game Config

Code	Comments
export const PlatformerConfig = {	
...BaseConfig,	// using config left card
physics: {	
default: 'arcade',	
arcade: {	
gravity: { y: 300 },	
debug: false	
}	
},	
};	

Create Phaser Game Object

Code
const withParentAndScene = {

```
  super('TemplateGame');
 }
 preload(){
  // loading image assets...
 }
 create(){
  // create game sprites
 }
 update(){
  // rendering changes loop
 }
}
```

Preload Game Assets

```
preload(){
 // load bullet image
 this.load.image('bullet', 'assets/sprites/bullet5.png');
 // define solder sprite
 const soldierInfo = [
  'soldier',
  'assets/sprites/solider_yellow.png',
  { frameWidth: 48, frameHeight: 48}
```

```
 ...PlatformerConfig,
 scene: [currentGame, Congratulations, GameFailed]
}
const game = new Phaser.Game(withParentAndScene)
```

Create Text function

```
/**
* create/update a text on screen
* @param {string} message text display
* @param {int} x horizontal position
* @param {int} y vertical position
*/
_createGuideText(message, x=10, y=10) {
 if (this.guideTxt) {
  this.guideTxt.removeFromDisplayList()
 }
 const params = [
  x, y, message, { fill: '#ffff00' }
 ]
 this.guideTxt = this.add.text(...params);
}
```

```
]
// load soldier sprite
this.load.spritesheet(...soldierInfo);
// load fire sound
this.load.audio('fire', 'assets/audio/blaster.mp3');
}
```

Listen Mouse and Keyboard

```
create(){
  // enable mouse click
  this.input.on('pointerdown', function() {
  // do something while click on game stage
  })
  // enable keyboard operation
  this.cursors = this.input.keyboard.createCursorKeys();
  // disable space key presss, conflict with monaco editor
  this.input.keyboard.removeCapture(32);
}
```

Shoot Enemy Detection

```
create(){
```

Create Explosion Animation

```
preload(){
  const explosion = [
    'boom', 'assets/sprites/explosion.png',
    { frameWidth: 64, frameHeight: 64, endFrame: 23 }
  ]
  this.load.spritesheet();
}
create(){
  const explode = {
    key: 'explode', frames: 'boom',
    frameRate: 36, hideOnComplete: true
  };
  this.anims.create(explode);
}
```

Player Move by Keyboard

```
update(){
  if (this.cursors.left.isDown) {
```

Code

```
const groupMeta = [[], {runChildUpdate: true}]
this.enemies = this.add.group(...groupMeta);
this.playerLasers = this.add.group(...groupMeta);
// collision detection
this.physics.add.collider(
  this.enemies, this.playerLasers,
  function(enemy, bullet){
    // explosion sound play
    // destroy enemy, bullet
  }, null, this
 )
}
```

Lazy todo Something

Code

```
create(){
 this.time.addEvent({
   delay: 200,
   callback() {
    // do something...
   },
```

Code

```
   // player walk left
   return
 } else if (this.cursors.right.isDown) {
   // player walk right
   return
 } else if (this.cursors.up.isDown){
   // player jump
   return
 }
}
```

Create Bouncing Text

Code

```
_createTextBouncing() {
 this.tweens.add({
   targets: this.guideTxt,
   y: 30,
   duration: 500,
   repeat: 4,
   paused: false,
   yoyo: true
 });
}
```

```
    callbackScope: this,
    loop: false,
  })
}
```

Create Player Sprite

```
preload(){
  this.load.spritesheet(
    'player',
    'assets/sprites/player_tilesheet.png',
    { frameWidth: 80, frameHeight: 110 }
  );
}
_createPlayer(){
  this.player = this.physics.add.sprite(20, 0, 'player', 0);
  this.player.setBounce(0.2);
  this.player.setScale(0.4, 0.4);
  this.player.setCollideWorldBounds(true);
  this.player.setDepth(1)
}
```

Create Tilemap and Layer

```
preload(){
  const tilesSrc = [
    'tiles', 'assets/tilemaps/tiles/kenney_redux_64x64.png'
  ]
  const mapSrc = [
    'map', 'assets/tilemaps/maps/lavaAdventure.json'
  ]
  this.load.image(...tilesSrc);
  this.load.tilemapTiledJSON(...mapSrc);
}
create(){
  var map = this.make.tilemap({ key: 'map' });
  var tiles = map.addTilesetImage('adventure', 'tiles');
  this.background = map.createLayer('background', tiles, 0, 0);
}
```

Create Player Animation

```
_createPlayerAnimation() {
```

Hit Exit Check

```
create(){
  // 2: wall, 13: grass block, 96: exit
  this.background.setCollision([2, 13,])
  const doorFilter = tile => tile.index === 96
  this.exits = this.background.filterTiles(doorFilter)
}
_hitExitCheck(){
  const overlapParams = [
    this.player, this.exits, this._hitExit, null, this
  ]
  this.physics.world.overlapTiles(overlapParams);
}
```

Close Enough Check

```
/**
* touch enough to consider a real hit
* @param {Sprite} player
* @param {Tile} tile
* @param {Number} distance
```

```
this.anims.create({
    key: 'turn',
    frames: [ { key: 'player', frame: 0 } ],
    frameRate: 6
});
const goLeft = ['player', { start: 10, end: 9 }]
this.anims.create({
    key: 'left',
    frames: this.anims.generateFrameNumbers(...goLeft),
    frameRate: 6,
    repeat: -1
});
const goRight = ['player', { start: 9, end: 10 }]
this.anims.create({
    key: 'right',
    frames: this.anims.generateFrameNumbers(...goRight),
    frameRate: 6,
    repeat: -1
  });
}
```

Code
* @returns true or false
*/
_closeEnough(player, tile, distance, tileWidth = 32) {
var tx = tile.x * tileWidth + tileWidth/2
var px = player.body.center.x
var horiDifference = Math.round(Math.abs(tx - px))
return distance > horiDifference
}