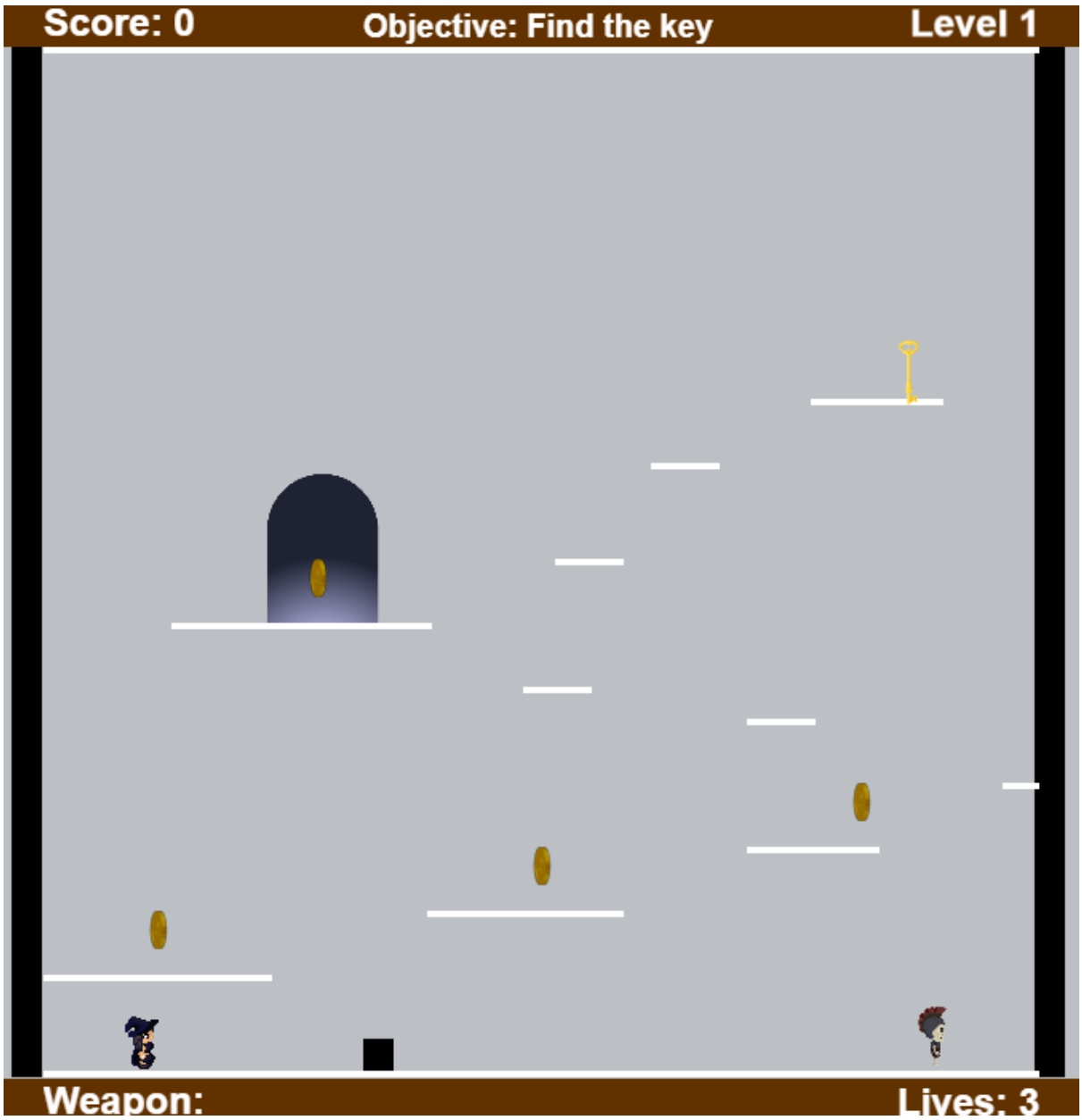
Multimedia Coursework 2

This project used Phasor.JS framework.

Objective of the game:

- Find the key and unlock the door.

Node.JS + Brackets IDE may need to be required to run on a local machine. If you are unable to run the game, go to:
http://users.sussex.ac.uk/~ss799 **. The games files will not be changed.**



Game Menu

A game menu has been implemented which is the first thing the user sees. If the click anywhere on the screen it will load the Level1 State, this is where Level 1 begins.

```
create: function(){
    game.add.tileSprite(30, 30, 650, 640, 'background')
    game.add.text(game.world.width*0.5, game.world.height*0.5, 'Start Game', { font: '18px Arial', fill: '#ffff'});


    console.log('You are in the Menu state');
    game.input.onDown.add(function(){
        changeState('Level1');
    });
}
```

Keyboard Control

The player is able to use the common gestures when playing a game. UP DOWN LEFT RIGHT to move/ jump around as well as SPACEBAR to shoot an enemy when picking up the weapon. This is done by:

```
// Move the player when an arrow key is pressed
if (this.cursor.left.isDown){
    this.player.body.velocity.x = speedLeft;
    this.player.scale.x = -1;
    facing = false;
}
else if(this.cursor.right.isDown){
```

This function here means that if the user presses the LEFT button (Is down) then move the player to the left with the specific velocity and flip the PNG image to -1 (facing left). Similarly for jumping if the cursor is UP then give the player the $velocity.y$ a value. (Velocity.y = moving the player up In the air by a certain distance).

Limited Movement using Collision Detection

This was done by using collision detection. If the player touches a block then it should stop the player from moving any further in that direction, AKA boundaries.

```
game.physics.arcade.collide(this.player, this.walls);
game.physics.arcade.collide(this.player, this.sidewalls);
game.physics.arcade.collide(this.player, this.collapsefloors);
```

If the player is collided with the wall object(An array of PNG images) then it sets the players velocity to 0 in any direction it is at.

Collision Detecting with enemies

Enemies itself are an array of PNG images. This uses the same principles as the boundary detection, however if the player is overlapped with the enemy in the wrong direction then it loses 1 life.

```
// kill enemy
game.physics.arcade.overlap(this.player, this.enemies, this.killEnemy, null, this);
```

This method here checks if the player is overlapped with the enemy. If it is, it calls KillEnemy.

```
killEnemy: function(player,enemies) {
   // jump to kill enemy
if(enemies.body.touching.up && player.body.touching.down){
    enemyHealth =- 100;
    // in this case just jump again
    this.player.body.velocity.y =  -200;
    enemies.kill();
    score+=10;
    scoreText.text = "Score: " + score;
}
else{


   this.lives();
}
},
```

If the player lands on top of the enemy then it kills the enemy. Else if it doesn't it land in that direction(or overlapped in that direction) then it reduces the players lives by 1.

Life System

The player starts of with 3 lives. If they land on an enemy (or touched by the enemy) or killed by the enemy then it reduces the players lives by 1. This function below, essentially checks if the players lives are greater than 0 after taking off the life. If it is, then it lets the player play that map again else It restarts the game back to Level 1.

```
lives: function(){
lives--;
    if(lives >= 0 ) {
        livesText.setText('Lives: '+lives);
        lifeLostText.visible = true;
        this.player.reset(100,600);
        game.input.onDown.addOnce(function(){
            lifeLostText.visible = false;
            this.player.visibility = true;
            this.player.reset(100,600);
            this.player.body.velocity.set(150, -150);
        }, this);
    }
    else {
       lives = 4;
       changeState("Level1");

    }
}
```

Scoring System

There are different scoring systems implemented.

- Collecting a coin: 1 point
- Killing an enemy by landing on top: 10 points
- Killing the enemy with a gun: 2 points
- Killing the boss: 50 points

The scoring system works by creating a text field and adding the integer value to the existing one.

```
score+=10;
scoreText.text = "Score: " + score;
```

The score starts off with 0. After the player performs an action to retrieve the points, then it increments the score by the specific value and concatenates it to the string.

Shooting Enemies

The player, can shoot enemies when collecting the fireball weapon. Once they have done this, each fireball shot towards to the enemy will be killed, if it's in the same direction as them.

```
fire :function(){
if(canFire){
      if (game.time.now > nextFire && bullets.countDead() > 0)
     {
          nextFire = game.time.now + fireRate;

          var bullet = bullets.getFirstDead();
          bullet.scale.setTo(0.05,0.05);

          bullet.reset(this.player.x,this.player.y + 8);
          console.log("Shooting");
          bullet.body.velocity.x = 400;

      }
  }
```

Assuming the user has picked up the fireball; a bullet is created every time they press the SPACEBAR. This takes in the PNG image used for shooting and is sent from the players x,y coordinates with a velocity of 400. When the bullet is overlapped with the enemy it calls collisionEnemyBullet.

```
collisionEnemyBullet: function(enemies,bullets) {
     bullets.kill();
     enemies.kill();
     if(direction == -1){
         console.log("Right enememy");
         enemies.kill();
     }
      if(direction == 1){
         console.log("Right enememy");
         enemies.kill();
     }
     score+=2;
     scoreText.text = "Score: " + score;

 },
```

In level 3, which is where the user can begin to shoot; it kills the enemies instantly. Enemies do not have health, however in the Boss level, the boss has 100HP and each bullet causes 15damage.

```
collisionEnemyBullet: function(bullets, enemies) {
      console.log("Killing");
      bullets.kill();
      enemyHealth -= 10;
      if(enemyHealth < 0){
          enemies.kill();
          score += 50;
          scoreText.text = "Score: " + score;
          changeState('GameOver');

      }
 },
```

PowerUps

Only 1 powerup is implemented, which is jump. If the player picks up the flight icon then it updaes the players movement variables, enabling it to “jump” higher.

```
takeFlight: function(player, flights) {
    flights.kill();
    jump = -515;
    powerFlight = game.add.text(300, 40, 'Super Jump Activated', { fontSize: '12px'});
},
```

Lasers

Lasers are implemented by creating a string PNG images. That line up to its X,Y coordinate. If the player is overlapped with the laser, then the enemy loses a life.

Lasers flicker between on and off every 1ms. To turn the lasers off the player needs to find the button and press enter on It.

```
disableLaser: function(player,buttons){
  if(!hovered){
      hovered = true;
      game.add.text(300, 40, 'Press Enter to disable the laser!', { fontSize: '12px'});
  }
  if(ENTER.isDown){
   if(on){
        this.lasers.destroy();
        on = false;
    }
  }


},
```

Level Creation

Levels are created by creating an array of the map. Each value in the map corresponds to an x,y coordinate where the image is mapped and scaled to.

```
   // Design the level. s = side wall x = wall, o = coin, ! = lava, d = door locked , do = door open, f = collapable floors,
   //1 = flight, p = protector, h laser hold, l laser, f = fireball
var level = [
       'xaaaaaaaaaaaaaaaaaaaaaaaaaaaaax',
       'x ooooooooooooohooooooooooooo  x',
       'x               l              x',
       'x    a  a  a    l a    a   a    x',
       'x            a  l              x',
       'x               l              x',
       'x               l     a        x',
       'x               l              x',
       'x               l              x',
       'x               l          a   x',
       'x               l      m   e   mx',
       'x               l              x',
       'x               l     aaaaaaaaax',
       'x               l              x',
       'x               l            k x',
       'x               l  m  e        mx',
       'x               l              x',
       'xm e            l aaaaaaaaaaaaax',
       'x           m   l              x',
       'xaaaaaaaaaaaaa  l              x',
       'x               l m     e     m x',
       'x               l              x',
       'x               l aaaaaaaaaaaa x',
       'xm    e      m  l              x',
       'x               l              x',
       'xaaaaaaaaaaaaa  l  m     e     mx',
       'x               l           d  x',
       'x m    e      m l  aaaaaaaaaaaax',
       'x               l              x',
       'x aaaaaaaaaaaaa l              x',
       'x            b l m      e     mx',
       'x2              l      k       x',
       ' aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa ',
   ];
// Create the level by going through the array
```

The code below is a snippet of how an image is loaded into the level where doorLocked corresponds to a PNG image that has been preloaded.

```
for (var i = 0; i < level.length; i++) {
    for (var j = 0; j < level[i].length; j++) {
         if (level[i][j] == 'd') {
            var dooropen = game.add.sprite(30+20*j, 578, 'doorlocked');
            dooropen.body.immovable = true;
            dooropen.scale.setTo(0.2,0.2);
             this.doors.add(dooropen);

        }
```

Level Progressing

To progress to the next level, the player must pick up the key and make their way to the door. Once they have done this, changeState() is called and loads in the next level.

```
openDoor: function(player, doors) {
 if(hasKey){
    changeState('Level3');
 }else
 {
 }

},


takeKey: function(player, key) {
    key.kill();
    hasKey = true;
    objective.text = "Objective: Unlock the door";

},
```

Boss Level

In this level, the boss can shoot the player.

Once the player has completed all other levels, they are sent to the Boss Level. The boss shoots bullets towards the players coordinates with a velocity of 300ms. This is the same implementation as the player shooting an enemy, but however with the enemy shooting towards a specific coordinate (MoveToObject) is the function that allows this.

```
 EnemyFire: function() {
      livingEnemies.length = 0;
        this.enemies.forEachAlive(function(enemy){
            livingEnemies.push(enemy)
        });

    if(this.time.now > enemyBulletTime) {
                enemyBullet = enemyBullets.getFirstExists(false);
                var shooter = livingEnemies[0];
                enemyBullet.reset(shooter.body.x, shooter.body.y + 30);
                enemyBulletTime = this.time.now + 500;
                this.physics.arcade.moveToObject(enemyBullet,this.player,300);
        }
},
```

References:

All images used are royalty free images/ developed myself apart from the ones referenced below

DeviantArt. (2018). *Pixel Sprite Boss Agreath*. [online] Available at: https://resa11.deviantart.com/art/Pixel-Sprite-Boss-Agreath-352531300 [Accessed 18 May 2018].

freeiconspng.com. (2018). *Fireball1 GD BlueKPL..> #2441 - Free Icons and PNG Backgrounds*. [online] Available at: https://www.freeiconspng.com/img/2441 [Accessed 18 May 2018].

OpenGameArt.org. (2018). *Rotating Crystal Animation (8-Step)*. [online] Available at: https://opengameart.org/content/rotating-crystal-animation-8-step [Accessed 18 May 2018].

Pinterest. (2018). *Game Art: Fx's elements*. [online] Available at: https://www.pinterest.com/pin/429601251932625874/ [Accessed 10 May 2018].