BONAFIDE CERTIFICATE

This is to certify that record of course work is a bonafide work done by **Ajil Pappachan**, ID No.: **2018UG03077**, in partial fulfillment of the requirements for the **2nd year B.Sc. Game Programming** during the academic year **2019 – 2020** is the original work of the candidate.

Submitted for the **WEB TECHNOLOGY-1** assessment held on _____.

_____ _____

Verified By **Staff In-Charge**

GAME DESIGN DOCUMENT

The Shapening

INDEX

INTRODUCTION

The Shapening is an endless single-player hyper-casual game, the game is inspired by subway surfers the android game and is intended to take a step back and make a web game based on the game's principles. It consists of 3 zones through which a player must navigate in order to obtain a high score, by dodging the colored expanding tile in time. UI consists of a timer and a scoreboard, and the objective of the player is to get a high score.

THEME

The theme of the objective game output is **Back**. And true to the theme, the game takes the concept of a modern 3D game like Subway Surfers and brought the gameplay mechanics and concept **back** to a simple, retro, 2D web game made up of simple shapes and colors.

GAME DESIGN

Game Name	:	The Shapening
Genre	:	2D Endless, Skill-based, Multiplayer, Hyper casual Web
Target Audience	:	Web Surfers, Teen
Target Platform	:	PC (Web Browsers)

SOFTWARE SPECIFICATIONS

Operating System: Microsoft Windows 10 Pro 64 bit

API: DirectX 12

HARDWARE SPECIFICATIONS

System model: Hp Omen

Processor: Intel® Core ™ i7-8500 CPU@2.50GHz

RAM Size: 16 GB

BUILT ENVIRONMENT

HTML Canvas (Scripting)

JavaScript (Scripting)

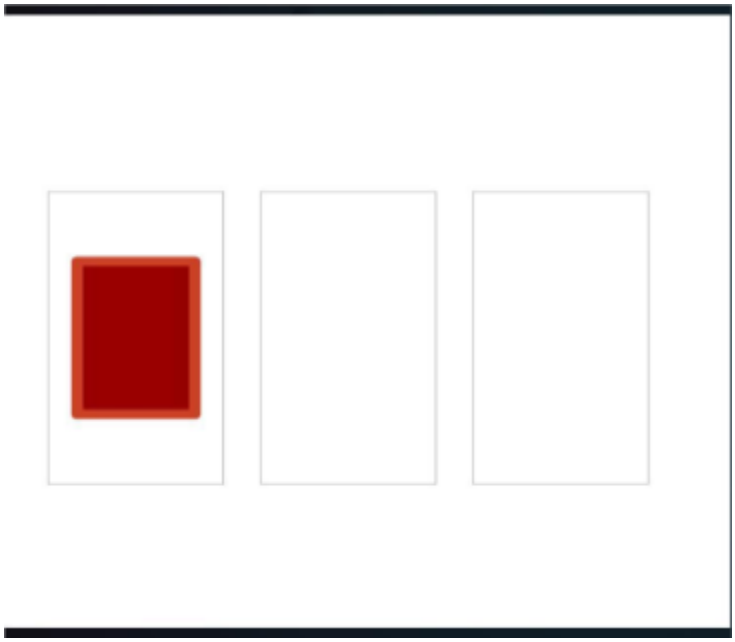
CSS (Scripting)

Visual studio code (VSC)

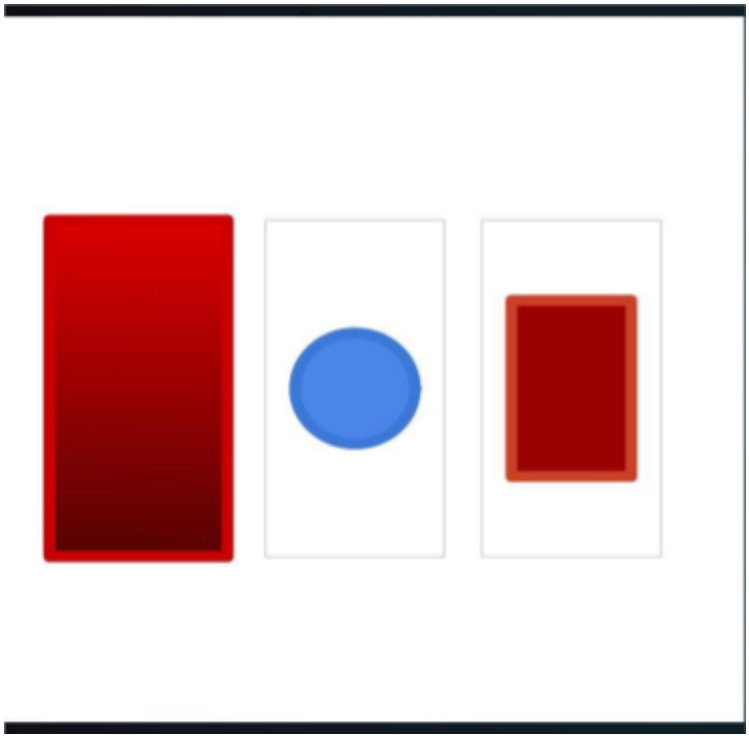
PITCH DESIGN

Level Design

(Fig. Game canvas layout)



(Fig. Zone)

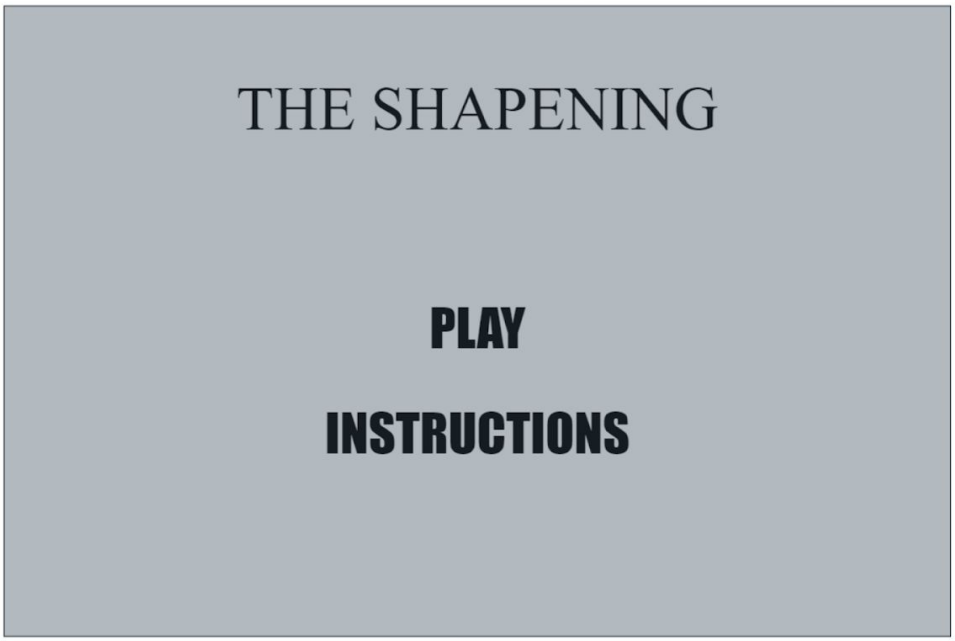


(Fig. The game)

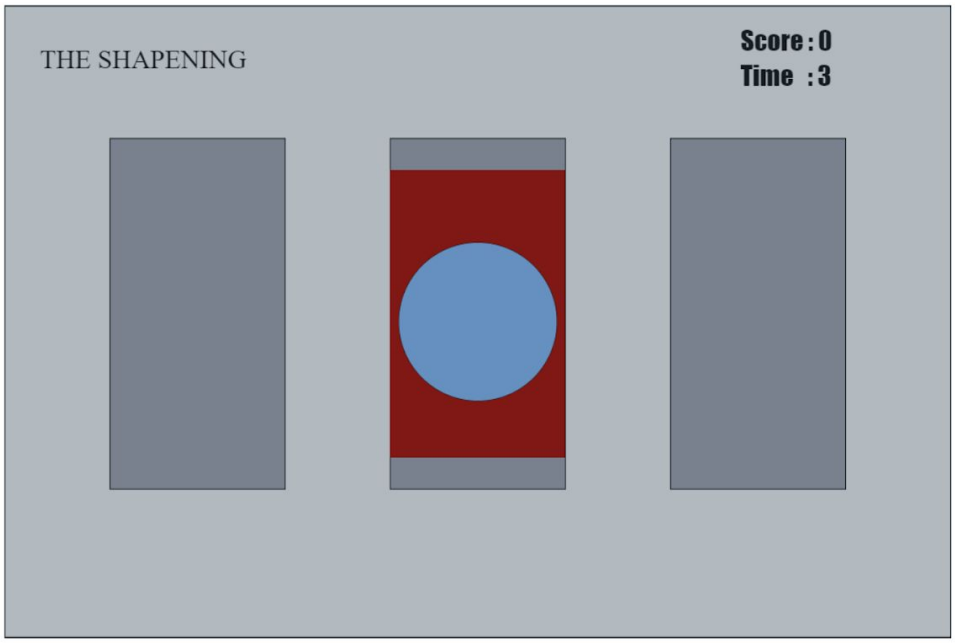
Character design

(Fig. Player)

FINAL GAME DESIGN



(Fig. Main menu)



(Fig. Gameplay)

(Fig. Game over screen)

GAMEPLAY

This game is an endless game, where the player has to avoid the red zones as shown above in order to progress further and achieve a high score.

Score

The player will get points based on the current difficulty level with time.
The difficulty level increases every 10 points and the game speeds up.

Mechanics

Collision- The player can collide with a Killzone

Motion – horizontal motion of the player

Time - Game difficulty increases with time

USER INTERFACE

The user can interact with the game using the keyboard and the Main Menu using the mouse.
The user interface consists of arrow keys for player control.

UI Controls:

Mouse for selecting an option in the Main Menu.
Arrow keys to control the player during gameplay.

GAME OPTIMIZATION

Since the game is followed by the rules of js13k competition, the game should be optimized to 13KB of size. No external libraries used to optimize the game. Visual studio code used to program the game. The entire game made using JavaScript canvas and minimal HTML and no CSS.

CODE SNIPPETS

HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>The Shapening</title>
    <style>
        canvas {
            border: 1px solid black;
            position: absolute;
            height: 100%;
            width: 100%;
        }

        body {
            margin:0%;
        }
    </style>
</head>
<body>
    <canvas></canvas>
    <script src = "GameScript.js"></script>
</body>
</html>
```

JavaScript:

```
//Initialization

var canvas = document.querySelector("canvas");

canvas.width = canvas.scrollWidth;
canvas.height = canvas.scrollHeight;

var ctx = canvas.getContext("2d");

ctx.strokeStyle = "#151C22";

var gameStart = false;
var isPlaying = false;
var isPaused = true;
```

```
var pauseTimeStart = 0;
var pauseTimeEnd = 0;
var isInstructing = false;
var backInstructing = false;

//Main Menu

function MainMenu() {
    this.heading = "THE SHAPENING";
    this.play = "PLAY";
    this.instructions = "INSTRUCTIONS";

    this.headingWidth;

    ctx.font = "60px Impact";
    this.playWidth = ctx.measureText(this.play).width;
    this.instructWidth = ctx.measureText(this.instructions).width;

    this.canPlay = false;
    this.canInstruct = false;

    this.drawMenu = function () {
        ctx.clearRect(0, 0, canvas.width, canvas.height);
        ctx.fillStyle  = "#B2B9BF";
        ctx.fillRect(canvas.width / 2 - 540, canvas.height / 2 - 360, 1080, 720);
        ctx.fillStyle = "#151C22";
        ctx.strokeRect(canvas.width / 2 - 540, canvas.height / 2 - 360, 1080, 720);

        ctx.font = "70px Times New Roman";
        this.headingWidth = ctx.measureText(this.heading).width;
        ctx.fillText(this.heading, (canvas.width / 2) - (this.headingWidth / 2), 150);

        ctx.font = "60px Impact";
        ctx.fillText(this.play, (canvas.width / 2) - (this.playWidth / 2), 400);

        ctx.font = "60px Impact";
        ctx.fillText(this.instructions, (canvas.width / 2) - (this.instructWidth / 2), 520);
    }
}
var mainMenu = new MainMenu();

//Instructions

function Instruction() {
    ctx.clearRect(0, 0, canvas.width, canvas.height);
    ctx.fillStyle  = "#B2B9BF";
    ctx.fillRect(canvas.width / 2 - 540, canvas.height / 2 - 360, 1080, 720);
    ctx.fillStyle = "#151C22";
    ctx.strokeRect(canvas.width / 2 - 540, canvas.height / 2 - 360, 1080, 720);
    ctx.font = "60px Impact";
    ctx.fillText("INSTRUCTIONS", canvas.width / 2 - mainMenu.instructWidth / 2, canvas.height
 / 2 - 300);
    this.instructions = ["You control the Circle shape.", "You can move left and right betwee
n the designated playzones.", "Each zone is outlined by a Rectangle shape.", "Zones gets fille
d up randomly.", "When a zone is completely filled by a Rectangle, the zone becomes a Killzone
```

```
.", "The Objective of the player is to avoid being in a Killzone.", "Game ends if the player
is in a Killzone.", "Score is calculated based on survival time."];
    ctx.font = "30px Impact";
    for (var i = 0; i < 6; i++) {
        ctx.fillText(instructions[i], canvas.width / 2 - 500, (canvas.height / 2 - 200) + (i
* 50), 1000);
    }
    ctx.font = "30px Times New Roman";
    ctx.fillText("Ajil Pappachan", canvas.width / 2 - 520, canvas.height / 2 + 350);
    ctx.fillText("Cheese Brain", canvas.width / 2 + 360, canvas.height / 2 + 350);
    ctx.fillText("Back", canvas.width / 2 - 20, canvas.height / 2 + 200);
    ctx.strokeRect(canvas.width / 2 - 30, canvas.height / 2 + 170, 80, 40);
}

//GAME!!

var difficultyLevel = 1;
var scoreToNextLevel = 1;

//Player

function Player() {
    this.centreX = canvas.width / 2;
    this.centreY = canvas.height / 2;
    this.radius = 90;
    this.startAngle = 0;
    this.endAngle = Math.PI * 2;

    this.isDead = false;

    this.update = function (newX, newY) {
        this.centreX = newX;
        this.centreY = newY;
    }

    this.draw = function () {
        ctx.fillStyle = "#6690C0";
        ctx.beginPath();
        ctx.arc(this.centreX, this.centreY, this.radius, this.startAngle, this.endAngle);
        ctx.stroke();
        ctx.fill();
        ctx.fillStyle = "#151C22";
    }
}
var player = new Player();

//UI

function UI() {
    this.title = "THE SHAPENING";
    this.score = 0;
    this.time = 0;
    this.gameStartTime;

    this.startGame = function () {
```

```
        this.gameStartTime = Date.now();
    }

    this.draw = function (GameTime) {
        var scoreMultiplier = difficultyLevel / 10;
        ctx.font = "30px Times New Roman";
        ctx.fillText(this.title, canvas.width / 2 - 500, 80);

        this.score = Math.floor(GameTime * scoreMultiplier);
        ctx.font = "30px Impact";
        ctx.fillText("Score : " + this.score, canvas.width / 2 + 300, 60);
        ctx.fillText("Time   : " + GameTime, canvas.width / 2 + 300, 100);

    }
}
var ui = new UI();

//Zone

function Zone(x) {
    this.x = x;
    this.y = 160;
    this.width = 200;
    this.height = 400;

    this.draw = function () {
        ctx.fillStyle = "#79818E";
        ctx.fillRect(this.x, this.y, this.width, this.height);
        ctx.strokeRect(this.x, this.y, this.width, this.height);
        ctx.fillStyle = "#151C22";
    }
}
var zone1 = new Zone(canvas.width / 2 - 420);
var zone2 = new Zone(canvas.width / 2 - 100);
var zone3 = new Zone(canvas.width / 2 + 220);

//Killzone

function KillZone()
{
    this.activeKillzone = false;
    this.maximumTime;
    this.minimumTime;
    this.timeToSpawn;
    this.zoneToSpawn;
    this.spawnX;
    this.spawnY = 160;
    this.newX = 0;
    this.newY = 0;
    this.width = 0;
    this.height = 0;
    this.speed = 2;

    this.activateKillzone = function (gameTime) {
        this.maximumTime = gameTime * (10 / difficultyLevel);
```

```
        this.minimumTime = gameTime;
        this.timeToSpawn = Math.random() * (this.maximumTime - this.minimumTime) + this.minim
umTime;
        this.zoneToSpawn = Math.floor(Math.floor(Math.random() * 3) + 1);
        this.activeKillzone = true;

        if(this.zoneToSpawn == 1)
            this.spawnX = Math.floor(canvas.width / 2 - 420);
        if(this.zoneToSpawn == 2)
            this.spawnX = Math.floor(canvas.width / 2 - 100);
        if(this.zoneToSpawn == 3)
            this.spawnX = Math.floor(canvas.width / 2 + 220);
    }

    this.drawKillZone = function () {

        if(this.width < 200)
            this.width += this.speed;
        if(this.height < 400)
            this.height += this.speed;

        this.newX = (this.spawnX + 100) - this.width / 2;
        this.newY = (this.spawnY + 200) - this.height / 2;
        ctx.fillStyle = "#801815";
        ctx.fillRect(this.newX, this.newY, this.width, this.height);
        ctx.fillStyle = "#151C22";
    }

    this.kill = function () {
        ctx.fillStyle = "red";
        ctx.fillRect(this.newX, this.newY, this.width, this.height);
        if(Math.floor(player.centreX) == Math.floor(this.spawnX + 100))
        {
            player.isDead = true;
            isPlaying = false;

        }
        this.width = 0;
        this.height = 0;
        this.activeKillzone = false;
    }
}
var killZone = new KillZone();

//Game Over

function gameOver () {
    isPlaying = false;
    ctx.clearRect(canvas.width / 2 - 540, canvas.height / 2 - 360, 1080, 720);
    ctx.fillStyle  = "#B2B9BF";
    ctx.fillRect(canvas.width / 2 - 540, canvas.height / 2 - 360, 1080, 720);
    ctx.fillStyle = "#151C22";
    ctx.strokeRect(canvas.width / 2 - 540, canvas.height / 2 - 360, 1080, 720);
    ctx.font = "60px Times New Roman";
    this.gameOverText = "GAME OVER"
```

```
    ctx.fillText(this.gameOverText, canvas.width / 2 - ctx.measureText(this.gameOverText).wid
th / 2, canvas.height / 2 - 100);
    this.scoreText = ui.score;
    ctx.font = "50px Impact";
    ctx.fillText("Score", canvas.width / 2 - ctx.measureText("Score").width / 2, canvas.heigh
t / 2);
    ctx.fillText(this.scoreText, canvas.width / 2 - ctx.measureText(this.scoreText).width / 2
, canvas.height / 2 + 100);

}

//Gameplay

function Animation() {
    if (isPlaying)
        requestAnimationFrame(Animation);
    else if (player.isDead)
        requestAnimationFrame(gameOver);
    else
        requestAnimationFrame(mainMenu.drawMenu());

    //ctx.clearRect(0, 0, canvas.width, canvas.height);
    ctx.clearRect(canvas.width / 2 - 540, canvas.height / 2 - 360, 1080, 720);
    ctx.fillStyle  = "#B2B9BF";
    ctx.fillRect(canvas.width / 2 - 540, canvas.height / 2 - 360, 1080, 720);
    ctx.fillStyle = "#151C22";
    ctx.strokeRect(canvas.width / 2 - 540, canvas.height / 2 - 360, 1080, 720);

    var currentTime = Date.now() - (pauseTimeEnd - pauseTimeStart);

    var GameTime = (currentTime - ui.gameStartTime) / 1000;

    ui.draw(Math.round(GameTime));

    if(ui.score == scoreToNextLevel)
    {
        scoreToNextLevel = scoreToNextLevel + 10;
        difficultyLevel = difficultyLevel + 1;
        killZone.speed += difficultyLevel;
    }

    zone1.draw();
    zone2.draw();
    zone3.draw();

    if(!killZone.activeKillzone)
    {
        killZone.activateKillzone(GameTime);
    }
    else
    {
        killZone.drawKillZone();
        if(killZone.width >= 200 && killZone.height >= 400)
        {
            killZone.kill();
```

```
            //killZone.clear();
        }
    }


    player.draw();
}

//Input

//Key Press

window.addEventListener("keydown", keyPressed, false);
function keyPressed(event) {
    if (event.keyCode == 37 && player.centreX - 200 > zone1.x)
        player.update(player.centreX - 320, player.centreY);
    if (event.keyCode == 39 && player.centreX + 200 < zone3.x)
        player.update(player.centreX + 320, player.centreY);
    if (event.keyCode == 27) {
        isPlaying = false;
        if (!isPaused) {
            pauseTimeStart = pauseTimeStart + Date.now()
            isPaused = true;
        }
    }
}

//Mouse Movement

window.addEventListener("mousemove", mouseMovement, false);

function Mouse() {
    this.x = undefined;
    this.y = undefined;
}

var mouse = new Mouse();

function mouseMovement(event) {
    mouse.x = event.x;
    mouse.y = event.y;

    if (mouse.x > canvas.width / 2 - mainMenu.playWidth / 2 && mouse.x < canvas.width / 2 + m
ainMenu.playWidth / 2 && mouse.y > 300 && mouse.y < 400 && isPaused)
        mainMenu.canPlay = true;
    else
        mainMenu.canPlay = false;

    if (mouse.x > canvas.width / 2 - mainMenu.instructWidth / 2 && mouse.x < canvas.width /2
+ mainMenu.instructWidth / 2 && mouse.y > 450 && mouse.y < 550 && isPaused)
        mainMenu.canInstruct = true;
    else
        mainMenu.canInstruct = false;
```

```
    if (mouse.x > canvas.width / 2 - 30 && mouse.x < canvas.width / 2 + 50 && mouse.y > canva
s.height / 2 + 170 && mouse.y < canvas.height / 2 + 210 && isPaused)
        backInstructing = true;
    else
        backInstructing = false;
}

//Mouse Input

window.addEventListener("mousedown", mousePressed, false);
function mousePressed(event) {
    if (mainMenu.canPlay) {
        if (!gameStart) ui.startGame();
        isPlaying = true;
        if (gameStart) {
            pauseTimeEnd = pauseTimeEnd + Date.now()
            isPaused = false;
        }
        gameStart = true;
        isPaused = false;
        Animation();
    }
    if (mainMenu.canInstruct) {
        isInstructing = true;
        Instruction();
    }
    if (isInstructing == true && backInstructing == true) {
        isInstructing = false;
        mainMenu.drawMenu();
    }
}

//Start Game

mainMenu.drawMenu();
```