Caloric Intensity

Technical design document

Logo

Description automatically generated

### Contents

[Executive Summary 3](#_TOC_250018)

[Game Overview 3](#_TOC_250017)

[Technical Summary 3](#_TOC_250016)

[Equipment 4](#_TOC_250015)

[Hardware 4](#_TOC_250014)

[Software 4](#_TOC_250013)

[Evaluation 5](#_TOC_250012)

[Game Engine 5](#_TOC_250011)

[Target Platform 5](#_TOC_250010)

[Scheduling 6](#_TOC_250009)

[Development Plan 6](#_TOC_250008)

[Milestones 7](#_TOC_250007)

[Updates, Maintenance & DLCs 7](#_TOC_250006)

[Work Environment 7](#_TOC_250005)

[Remote Collaboration 7](#_TOC_250004)

[File Formats & Naming convention 8](#_TOC_250003)

[Levels 9](#_TOC_250002)

[Level 1 9](#_TOC_250001)

[Asset List 9](#_TOC_250000)

[Time Complexity-Codes 10](#_TOC_250002)

[PanelHandeler.cs 10](#_TOC_250001)

[CharacterSelection.cs 26](#_TOC_250000)

[LeaderBoardHandler.cs 30](#_TOC_250000)

[Player\_Controller.cs 31](#_TOC_250000)

#### Game Development Team

PRODUCER

Arman Shahriari

Chase Anzalone

Joaquin IFN

Peter Estrada Ledesma

PRODUCTION MANAGER

Arman Shahriari

Chase Anzalone

Joaquin IFN

Peter Estrada Ledesma

PRODUCTION COORDINATOR

Arman Shahriari

Chase Anzalone

Joaquin IFN

Peter Estrada Ledesma

GAME DESIGNERS

Joaquin IFN

Chase

Arman

Peter

SYSTEMS/IT COORDINATOR

Joaquin IFN

PROGRAMMERS

Arman Shahriari

Chase Anzalone

Joaquin IFN

Peter Estrada Ledesma

TECHNICAL ARTISTS

Joaquin IFN

Chase Anzalone

AUDIO ENGINEERS

Peter Estrada Ledesma

Chase Anzalone

UX TESTERS

Arman Shahriari

Joaquin IFN

# Executive Summary

## Game Overview

The object of the game is to feed the hungry people throughout the town of Crunchville. The game is arcade-style high score based, meaning that you will play the same scenario trying to achieve the fastest time and best score possible. Having limited amounts of ammunition (burgers and hotdogs), you must strategize an optimal path around the town to the hungry citizens while managing to restock your weaponry at the local burger joint as fast-food as possible. You are given a Burger-Blaster and a Dog-Dealer, the two fastest meal dealers in the retro verse.

Dishing out burgers and dogs has never felt so good! You won’t be left hungry in My Neighborhood!

## Technical Summary

Caloric Intensity is developed in approximately 3 weeks by roughly 4 people using the Unity game engine. For visual graphics, Piskel and FireAlpaca is used. For the sound creation we used Beep Box.

The game will be deployed for PC.

The minimum requirements include:

PC, MAC AND LINUX STANDALONE

OS: Windows XP SP2+, Mac OS X 10 .8+, Ubuntu 12 .04+, SteamOS+ Graphics card: DX9 (shader model 2 .0) capabilities; generally everything made since 2004 should work.

# Equipment

## Hardware

Members of the team used 2 15” MacBook Pros, and 2 HP pavilions laptops for game development and asset creation. All owned by team members prior to start of the project.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PRODUCT | TASK | COST\* | QUANTITY | TOTAL |
| MacBook Pro 15” | Asset Creation Development  Visual Graphics | $0 .00 | 2 | $0.00 |
| HP pavilions | Asset Creation Development Visual Graphics | $0 .00 | 2 | $0 .00 |
| *\*values listed are general approximations in USD* | | | TOTAL | $0.00 |

## Software

All the software used for the development of *Zombie Toys* will be able to produce high end visuals, while still being able to deploy across different platforms. Not all team members will utilize all software tools. Software requirements and selections will vary based on team member roles and responsibilities.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PRODUCT | TASK | COST\* | QUANTITY | TOTAL |
| Unity3D | Game Editor /Engine | $0 .00 | 1 | $0 .00 |
| Piskel | Visual Graphics | $0 .00 | 1 | $0 .00 |
| FireAlpaca | Visual Graphics | $0 .00 | 1 | $0 .00 |
| Beep Box | Sound creation | $0 .00 | 1 | $0 .00 |
| *\*values listed are general approximations in USD* | | | TOTAL | $0.00 |

# Evaluation

## Game Engine

We used the Unity in the development of our game which provides a simple solution for creating 3d game with many users around the worlds. Benefits of using this game engine is being supported by a large community of developers, and great online tutorials which was the best solution for us to create our games.

## Target Platform

*Coloric Intensity will be deployed for PC which is great option for new game developers of Unity to start with.*

# Scheduling

## Development Plan

## This is approximately how our plan was, and we could complete many of these plans on the right date

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PRODUCT | 25 NOVEMBER 2021 | 30 NOVEMBER 2021 | 2 DECEMBER 2021 | 6 DECEMBER 2021 |
| 2D Art | Main menu visual graphics  Logo |  | HUD prototype  Inventory System | Final Hud and game Inventory design for PC |
| 3D Assets | Kenny Models | Adding Kenny Mode | Adding Kenny Models  To the game and creating game environment. | Soda model and animation |
| Scripting | - | PlayerControl  Main Menu  Bullet Handler  Player Collision  Controller  PanelHandler  InventoryIntem | - | Credit scene  Sound Controller  Character Selection  Splash activity scene  LeaderBoard Scene |
| Audio | - | - | Main soundtrack | Drinking soda sound  Customer satisfaction  Shooting sound |

## Milestones

|  |  |
| --- | --- |
| 25 Nov 2021 | Finding Appropriate models and creating visual graphics |
| 30 Nov 2021 | Implementing Character Main functionalities |
| 2 Dec 2021 | Completing the Game environment |
| 6 Dec 2021 | Adding HUD, Soda animation, sound  and leader board scene to the game |

## Updates, Maintenance & DLCs

Most of the things went as planned. We only decided some of the game instructions

to the main scene

|  |  |  |
| --- | --- | --- |
| 6 Dec 2021 | Game Instruction | Adding game instruction to the game |

# Work Environment

## Remote Collaboration

We are team of four students developing this game with very busy schedule. So we decided to work remotely, and communicate through Zoom, Discord, and getting use of google docs to communicate and work with each other remotely.

# File Formats & Naming Convention

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ASSET TYPE | SUBTYPE | NAMING CONVENTION | FILE FORMAT | ANNOTATIONS |
| 3D Asset | Main Player  Characters | Player  NPCNumber | FBX |  |
| Props | PropName | FBX |  |
| Environment | EnvironmentFistrict | FBX |  |
| Animations |  | AnimationName or movement |  | Spinning  Different Movements |
| Scripts | Player,  Panel,  Bullet Collision | NameController | C# | Weapon: Frost, Lightning, Slime, Stink  Attack: Attack, Debuff, Bolt, Projectile, Hit |
| Characters | CharacterNameBehavior | C# |  |
| Player | PlayerBehavior | C# |  |
| Materials | Colors | ColorName | \* .mat | Materials do not have a specific naming conven- tion in this project |
|  |  |  |  |
| UI/Graphics |  | Element Name | TGA PNG |  |

# Levels

## Level 1

The game environment for game of Coloric Intensity is a City with towers, building, shops, and a campsite where player is supposed to look for different hungry customers in the city and feed them in the time that is specified. If the player can’t find the customers in the city and feed them, he/she will lose the game.The hungry customers are located in the campsite, in front of the building, and fountain area. Hunger customers have different behaviors and animations.

A picture containing text

Description automatically generated

## Asset List

|  |  |
| --- | --- |
| Players | Customers |
| Enemies | None |
| Allies | None |
| Props | Bench  HotDog  Burger  Fries  Soda  Etc … |
| Environment | Districts:  Food,  Business, City,  Neighborhood  Others:  Trees, Campsites |

# Time complexity-Codes

# **I provided the code complexity in a separate Excel file as Professor Price mentioned to me during office hours. Here, I include the portion of my code for Caloric Intensity.**

# **PanelHandler.cs**

# **public class PanelHandler : MonoBehaviour**

# **{**

# **private GameObject Timer;**

# **private GameObject Customers;**

# **private GameObject HUDHolder;**

# **private Image HUDImage;**

# **private GameObject InventorySystem;**

# **private Image InventorySystemImage;**

# **private Image InventorySystemBurgerAmountImage;**

# **private Text InventorySystemBurgerAmountValue;**

# **private Image InventorySystemSpeedUpImage;**

# **private Text InventorySystemSpeedUpValue;**

# **private Image InventorySystemHotDogAmountImage;**

# **private Text InventorySystemHotDogAmountValue;**

# **private GameObject gameOverPanel;**

# **private GameObject WinningScreen;**

# 

# **private Button gameOverReplayBtn;**

# **private Button GameOverbackBtn;**

# **private Button WinningScreenReplayBtn;**

# **private Button WinningScreenBackBtn;**

# **private Text RemainingTime;**

# **private Text RemainingCustomers;**

# **private Text PlayerName;**

# **private int minute;**

# **private int seconds;**

# **private float totalTime;**

# **private bool occuring;**

# **private Text score;**

# **[SerializeField] Sprite Soda;**

# **private bool AlreadyEnded;**

# **// Start is called before the first frame update**

# **private GameObject gameInformation; // refrences the canvas to display the game information**

# **private bool informationActive; // boolean to know how the user is toggling the panel**

# **private AudioSource onDrink; //referneces drinking audio source FOUND IN EMPTY OBJECT UNDER CAMERA < AIM CONTROLLER**

# **private void Awake()**

# **{**

# **startNewGameFunctionality();**

# **InitialiseWinningScreenAndButtons();**

# **InitialiseGameOverScreenAndButtons();**

# **InitialiseInventorySystemScreenAndButtons();**

# **InitialiseHUDTextAndButtons();**

# **CalculateRemainingTime();**

# 

# **}**

# 

# **void Start()**

# **{**

# **updatePlayerName();**

# **gameReset();**

# **}**

# **/\*\***

# **\* Resetd the game for playing again**

# **\*/**

# **private void gameReset()**

# **{**

# **Time.timeScale = 1;**

# **occuring = false;**

# **Cursor.lockState = CursorLockMode.Locked;**

# **GameController.GameInstance.GainedSpeedUps = 0;**

# **GameController.GameInstance.playerSpeed = 10f;**

# **if (GameController.GameInstance.itemList.Count > 2)**

# **{**

# **GameController.GameInstance.itemList.RemoveAt(GameController.GameInstance.itemList.Count - 1);**

# **}**

# **onDrink = GameObject.Find("OnDrink\_Audio").GetComponent<AudioSource>(); //referneces audio clip with ondrink effect**

# **gameInformation = GameObject.Find("Canvas\_Information"); // Get reference to the canvas gameobject**

# **informationActive = false; // Ste boolean to false because initially we dont want it shwoing**

# **gameInformation.SetActive(informationActive); // Sets canvas to not visible**

# **}**

# **// Update is called once per frame**

# **void Update()**

# **{**

# **UpdateInventory();**

# **UpdateCustomer();**

# **GameWinnerFunctionality();**

# **GameOverConditionAndTimeFuctionality();**

# **OpenCloseHudeAndInventorySystem();**

# 

# 

# **}**

# **/\*\***

# **\* Shows player name on the screen**

# **\*/**

# **private void updatePlayerName()**

# **{**

# **if (GameController.GameInstance.characterName != null)**

# **{**

# **PlayerName.text = "Welcome " + GameController.GameInstance.characterName;**

# **}**

# **}**

# **/\*\***

# **\* If the player feeded require customers,**

# **\* the game will end and winning message shows up**

# **\* and its score will be calculated and shown on the screen**

# **\*/**

# **private void GameWinnerFunctionality()**

# **{**

# **if (GameController.GameInstance.numberOfCustomers <= 0)**

# **{**

# **if (!AlreadyEnded)**

# **{**

# **AlreadyEnded = true;**

# **Cursor.lockState = CursorLockMode.None;**

# **updateTopScorere();**

# **Time.timeScale = 0;**

# **WinningScreen.gameObject.SetActive(!WinningScreen.gameObject.activeInHierarchy);**

# 

# **}**

# **}**

# **}**

# **/\*\***

# **\* Update Top score name, and score**

# **\*/**

# **private void updateTopScorere()**

# **{**

# **int playerScored = (100 \* (int)totalTime);**

# **score.text = "Score: " + playerScored.ToString();**

# **if (GameController.GameInstance.gameDifficulty == "Easy")**

# **{**

# **if (playerScored > GameController.GameInstance.HighestScore[0])**

# **{**

# **GameController.GameInstance.HighestScore[0] = playerScored;**

# **GameController.GameInstance.topPlayer[0] = GameController.GameInstance.characterName;**

# **}**

# **}**

# **else if (GameController.GameInstance.gameDifficulty == "Medium")**

# **{**

# **if (playerScored > GameController.GameInstance.HighestScore[1])**

# **{**

# **GameController.GameInstance.HighestScore[1] = playerScored;**

# **GameController.GameInstance.topPlayer[1] = GameController.GameInstance.characterName;**

# **}**

# **}**

# **else if (GameController.GameInstance.gameDifficulty == "Hard")**

# **{**

# **if (playerScored > GameController.GameInstance.HighestScore[2])**

# **{**

# **GameController.GameInstance.HighestScore[2] = playerScored;**

# **GameController.GameInstance.topPlayer[2] = GameController.GameInstance.characterName;**

# **}**

# **}**

# **}**

# **/\*\***

# **\* Checking to see if game over**

# **\* If it is it shows game over screen**

# **\* stops the time,**

# **\* and let player decide to play again or not**

# **\*/**

# **private void GameOverConditionAndTimeFuctionality()**

# **{**

# **if (GameController.GameInstance.gameTime >= 0)**

# **{**

# **GameController.GameInstance.gameTime -= Time.deltaTime; ;**

# **totalTime = GameController.GameInstance.gameTime;**

# **CalculateRemainingTime();**

# **}**

# **else**

# **{**

# **if (!AlreadyEnded)**

# **{**

# **AlreadyEnded = true;**

# **Cursor.lockState = CursorLockMode.None;**

# **Time.timeScale = 0;**

# **gameOverPanel.gameObject.SetActive(!gameOverPanel.gameObject.activeInHierarchy);**

# **}**

# **}**

# **}**

# **/\*\***

# **\* Handle the case when player wants to close the inventory system**

# **\*/**

# **private void OpenCloseHudeAndInventorySystem()**

# **{**

# **if (Input.GetKeyDown(KeyCode.V))**

# **{**

# **HUDHolder.gameObject.SetActive(!HUDHolder.gameObject.activeInHierarchy);**

# **}**

# **if (Input.GetKeyDown(KeyCode.B))**

# **{**

# **InventorySystem.gameObject.SetActive(!InventorySystem.gameObject.activeInHierarchy);**

# **}**

# **if (Input.GetKeyDown(KeyCode.C))**

# **{**

# **GameController.GameInstance.GainedSpeedUps--;**

# 

# **if (GameController.GameInstance.GainedSpeedUps > -1 && occuring == false)**

# **{**

# **onDrink.Play();**

# 

# **StartCoroutine("speedupEffects");**

# **}**

# **}**

# **if (Input.GetKeyDown("i")) // Displays the information panel to the player if 'i' is pressed, if pressed again it will take it down**

# **{**

# **gameInformation.SetActive(!informationActive); // Makes the panel appear and disappear according to the last toggle.**

# **informationActive = !informationActive; // Changes the true and false according to the last toggle. This way it ensures that it will be taken down and up**

# **} // as user pleases**

# **}**

# **/\***

# **\* Handels the speedup soda effect functionality**

# **\* Gives 3x speed to player for 20 seconds**

# **\*/**

# **private IEnumerator speedupEffects(){**

# 

# 

# **occuring = true;**

# **GameController.GameInstance.playerSpeed \*= 2;**

# **yield return new WaitForSeconds(15f);**

# **GameController.GameInstance.playerSpeed /= 2;**

# **occuring = false;**

# **}**

# **/\*\***

# **\* Updates Time on the HUD**

# **\*/**

# **private void CalculateRemainingTime()**

# **{**

# **minute = (int)totalTime / 60;**

# **seconds = (int)totalTime % 60;**

# **if (seconds < 10)**

# **{**

# **RemainingTime.text = "Remaining Time: " + minute + ":0" + seconds;**

# **}**

# **else**

# **{**

# **RemainingTime.text = "Remaining Time: " + minute + ":" + seconds;**

# **}**

# 

# **}**

# **/\*\***

# **\* Updating number of cutomers**

# **\*/**

# **private void UpdateCustomer()**

# **{**

# **RemainingCustomers.text = "Remaining Customers: " + (GameController.GameInstance.numberOfCustomers).ToString();**

# **}**

# **/\*\***

# **\* Initialising elements on game Over screen/Panel**

# **\*/**

# **private void InitialiseGameOverScreenAndButtons()**

# **{**

# **if (gameOverPanel == null)**

# **{**

# **gameOverPanel = GameObject.Find("GameOverPanel").gameObject;**

# **gameOverReplayBtn = gameOverPanel.transform.Find("Replay").GetComponent<Button>();**

# **gameOverReplayBtn.onClick.AddListener(delegate { replayTheGame(gameOverPanel); });**

# **GameOverbackBtn = gameOverPanel.transform.Find("Back").GetComponent<Button>();**

# **GameOverbackBtn.onClick.AddListener(delegate { LoadSceneByName("Scene\_Menu"); });**

# **gameOverPanel.gameObject.SetActive(!gameOverPanel.gameObject.activeInHierarchy);**

# **}**

# 

# **}**

# **/\*\***

# **\* Initialising elements on Winning screen/Panel**

# **\*/**

# **private void InitialiseWinningScreenAndButtons()**

# **{**

# 

# **if(WinningScreen == null)**

# **{**

# **WinningScreen = GameObject.Find("WonPanel").gameObject;**

# **score = WinningScreen.transform.Find("Score").GetComponent<Text>();**

# **WinningScreenReplayBtn = WinningScreen.transform.Find("Replay").GetComponent<Button>();**

# **WinningScreenReplayBtn.onClick.AddListener(delegate { replayTheGame(gameOverPanel); });**

# **WinningScreenBackBtn = WinningScreen.transform.Find("Back").GetComponent<Button>();**

# **WinningScreenBackBtn.onClick.AddListener(delegate { LoadSceneByName("Scene\_Menu"); });**

# **WinningScreen.gameObject.SetActive(!WinningScreen.gameObject.activeInHierarchy);**

# **}**

# 

# **}**

# **/\*\***

# **\* Initialising elements on HUD Panel**

# **\*/**

# **private void InitialiseHUDTextAndButtons()**

# **{**

# **HUDHolder = GameObject.Find("HUDPanel").gameObject;**

# **Timer = HUDHolder.transform.Find("Timer").gameObject;**

# **Customers = HUDHolder.transform.Find("Customers").gameObject;**

# **HUDImage = HUDHolder.transform.GetComponent<Image>();**

# **HUDImage.color = GameController.GameInstance.HUDColor;**

# **PlayerName = HUDHolder.transform.Find("Name").GetComponent<Text>();**

# **RemainingTime = Timer.transform.Find("RemainingTime").GetComponent<Text>();**

# **RemainingCustomers = Customers.transform.Find("RemainingCustomers").GetComponent<Text>();**

# 

# 

# 

# **}**

# **/\*\***

# **\* Initialising elements on enventory system screen/Panel**

# **\*/**

# **private void InitialiseInventorySystemScreenAndButtons()**

# **{**

# **InventorySystem = GameObject.Find("GameInventoryPanel").gameObject;**

# **InventorySystemImage = InventorySystem.transform.GetComponent<Image>();**

# **InventorySystemImage.color = new Color32(GameController.GameInstance.redColor, GameController.GameInstance.greenColor, GameController.GameInstance.blueColor, 255);**

# **InventorySystemSpeedUpImage = InventorySystem.transform.Find((3).ToString()).GetComponent<Image>();**

# **InventorySystemSpeedUpValue = InventorySystemSpeedUpImage.transform.Find((3+"T").ToString()).GetComponent<Text>();**

# **InventorySystemSpeedUpValue.text = "";**

# **InventorySystemSpeedUpImage.enabled = false;**

# **InventorySystemBurgerAmountImage = InventorySystem.transform.Find((1).ToString()).GetComponent<Image>();**

# **InventorySystemBurgerAmountValue = InventorySystemBurgerAmountImage.transform.Find((1 + "T").ToString()).GetComponent<Text>();**

# **InventorySystemHotDogAmountImage = InventorySystem.transform.Find((2).ToString()).GetComponent<Image>();**

# **InventorySystemHotDogAmountValue = InventorySystemHotDogAmountImage.transform.Find((2 + "T").ToString()).GetComponent<Text>();**

# **}**

# 

# **/\*\***

# **\* It adds speedups to the enventory system**

# **\*/**

# **private void UpdateInventory()**

# **{**

# **if (GameController.GameInstance.GainedSpeedUps > 0 && GameController.GameInstance.GainedSpeedUps < 5)**

# **{**

# **for (int i = 0; i < GameController.GameInstance.itemList.Count; i++)**

# **{**

# **if (GameController.GameInstance.itemList[i].name == "Speedups")**

# **{**

# **InventorySystemSpeedUpImage.enabled = true;**

# **// set image to speedups**

# **InventorySystemSpeedUpImage.sprite = Soda;**

# **// update the text**

# **InventorySystemSpeedUpValue.text = (GameController.GameInstance.itemList[i].count).ToString();**

# **}**

# **}**

# **}**

# **InventorySystemBurgerAmountValue.text = GameController.GameInstance.GunBurgerAmount.ToString();**

# **InventorySystemHotDogAmountValue.text = GameController.GameInstance.GunHotDogAmount.ToString();**

# **if (GameController.GameInstance.GainedSpeedUps <= 0)**

# **{**

# **InventorySystemSpeedUpValue.text = "";**

# **GameController.GameInstance.GainedSpeedUps = 0;**

# **if (GameController.GameInstance.itemList.Count > 2)**

# **{**

# **GameController.GameInstance.itemList.RemoveAt(GameController.GameInstance.itemList.Count-1);**

# **InventorySystemSpeedUpImage.enabled = false;**

# **}**

# **}**

# **else**

# **{**

# **InventorySystemSpeedUpValue.text = GameController.GameInstance.GainedSpeedUps.ToString();**

# **}**

# 

# **}**

# **/\*\***

# **\* Doing appropriate reseting for replaying the game**

# **\*/**

# **private void replayTheGame(GameObject panel)**

# **{**

# **GameController.GameInstance.GainedSpeedUps = 0;**

# **InventorySystemSpeedUpValue.text = "";**

# **GameController.GameInstance.itemList.RemoveAt(GameController.GameInstance.itemList.Count - 1);**

# **InventorySystemSpeedUpImage.sprite = null;**

# **InventorySystemSpeedUpImage.enabled = false;**

# **panel.gameObject.SetActive(!panel.gameObject.activeInHierarchy);**

# **SceneManager.LoadScene("Scene\_Chase");**

# **}**

# **/\***

# **\* It loads the menu scene**

# **\*/**

# **public void LoadSceneByName(string sceneName)**

# **{**

# **GameController.GameInstance.GainedSpeedUps=0;**

# **InventorySystemSpeedUpValue.text = "";**

# **GameController.GameInstance.itemList.RemoveAt(GameController.GameInstance.itemList.Count - 1);**

# **InventorySystemSpeedUpImage.sprite = null;**

# **InventorySystemSpeedUpImage.enabled = false;**

# **SceneManager.LoadScene(sceneName);**

# **}**

# **/\***

# **\* It reset the game for a new game comming from character selection screen**

# **\*/**

# **private void startNewGameFunctionality()**

# **{**

# 

# **if (GameController.GameInstance.numberOfCustomers <= 0)**

# **{**

# **GameController.GameInstance.playerSpeed = 10f;**

# **GameController.GameInstance.GainedSpeedUps = 0;**

# **if (GameController.GameInstance.gameDifficulty == "Easy") { GameController.GameInstance.gameTime = 240; GameController.GameInstance.numberOfCustomers = 8; }**

# **if (GameController.GameInstance.gameDifficulty == "Medium") { GameController.GameInstance.gameTime = 210; GameController.GameInstance.numberOfCustomers = 10; }**

# **if (GameController.GameInstance.gameDifficulty == "Hard") { GameController.GameInstance.gameTime = 180; GameController.GameInstance.numberOfCustomers = 12; }**

# **AlreadyEnded = false;**

# **}**

# **if (GameController.GameInstance.gameTime <= 0)**

# **{**

# **GameController.GameInstance.GainedSpeedUps = 0;**

# **GameController.GameInstance.playerSpeed = 10f;**

# **if (GameController.GameInstance.gameDifficulty == "Easy") { GameController.GameInstance.gameTime = 240; GameController.GameInstance.numberOfCustomers = 8; }**

# **if (GameController.GameInstance.gameDifficulty == "Medium") { GameController.GameInstance.gameTime = 210; GameController.GameInstance.numberOfCustomers = 10; }**

# **if (GameController.GameInstance.gameDifficulty == "Hard") { GameController.GameInstance.gameTime = 180; GameController.GameInstance.numberOfCustomers = 12; }**

# **AlreadyEnded = false;**

# **totalTime = GameController.GameInstance.gameTime;**

# **}**

# **GameController.GameInstance.GunHotDogAmount = 12;**

# **GameController.GameInstance.GunBurgerAmount = 12;**

# **}**

# **}**

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Character\_Selection.cs

public class Character\_Selection : MonoBehaviour

{

private Button backBtn;

private Button quitBtn;

private Button makeCharacterBtn;

private InputField characterName;

private Dropdown gameDifficultyDropDownMenu;

private Dictionary<string, string> gameDifficultyHolder;

private Text gameDifficultyDesc;

private Slider redSlider;

private Slider blueSlider;

private Slider greenSlider;

private Image characterColor;

private GameObject colors;

// Start is called before the first frame update

void Start()

{

initiliseButtons();

initiliseCharacterNameInputField();

initilisegameDifficultyDropDownMenu();

initialiseSliders();

ReseGameContoller();

}

/\*\*

\* Reseting game data

\*/

private void ReseGameContoller()

{

GameController.GameInstance.gameDifficulty = "Easy";

GameController.GameInstance.gameTime = 240;

GameController.GameInstance.HUDColor = new Color32(0, 0, 0, 255);

GameController.GameInstance.redColor = 0;

GameController.GameInstance.blueColor = 0;

GameController.GameInstance.greenColor = 0;

GameController.GameInstance.GainedSpeedUps = 0;

GameController.GameInstance.numberOfCustomers = 8;

GameController.GameInstance.characterName = "";

characterColor.color = new Color32(0, 0, 0, 255);

GameController.GameInstance.numberOfCustomers = 8;

}

/\*\*

\* Initialising the sliders

\*/

private void initialiseSliders()

{

colors = GameObject.Find("CharacterColor").gameObject;

redSlider = colors.transform.Find("RedColor").GetComponent<Slider>();

greenSlider = colors.transform.Find("GreenColor").GetComponent<Slider>();

blueSlider = colors.transform.Find("BlueColor").GetComponent<Slider>();

Image playerColor = colors.transform.Find("CharacterColor").GetComponent<Image>();

characterColor = playerColor.transform.GetComponent<Image>();

redSlider.onValueChanged.AddListener(delegate { onChangeRedSliderValue(); });

greenSlider.onValueChanged.AddListener(delegate { onChangeGreenSliderValue(); });

blueSlider.onValueChanged.AddListener(delegate { onChangeBlueSliderValue(); });

}

/\*\*

\* Checks for change of value in the red slider

\*/

public void onChangeRedSliderValue()

{

GameController.GameInstance.redColor = (byte)redSlider.value;

changeImageColor();

}

/\*\*

\* Checks for change of value in the green slider

\*/

public void onChangeGreenSliderValue()

{

GameController.GameInstance.greenColor = (byte)greenSlider.value;

changeImageColor();

}

/\*\*

\* Checks for change of value in the blue slider

\*/

public void onChangeBlueSliderValue()

{

GameController.GameInstance.blueColor = (byte)blueSlider.value;

changeImageColor();

}

/\*\*

\* Updates image color and game data color

\*/

private void changeImageColor()

{

GameController.GameInstance.HUDColor = new Color32(GameController.GameInstance.redColor,

GameController.GameInstance.greenColor, GameController.GameInstance.blueColor, 255);

characterColor.color = new Color32(GameController.GameInstance.redColor,

GameController.GameInstance.greenColor, GameController.GameInstance.blueColor, 255);

}

/\*\*

\* Initialises game difficulty drop down menu

\*/

private void initilisegameDifficultyDropDownMenu()

{

gameDifficultyDropDownMenu = GameObject.Find("GameDifficultyDropdown").GetComponent<Dropdown>();

gameDifficultyDropDownMenu.options.Clear();

gameDifficultyHolder = new Dictionary<string, string>();

gameDifficultyHolder.Add("Easy", "To win, you'll have 240 seconds to feed 8 customers.");

gameDifficultyHolder.Add("Medium", "To win, you'll have 210 seconds to feed 10 customers.");

gameDifficultyHolder.Add("Hard", "To win, you'll have 180 seconds to feed 12 customers.");

gameDifficultyDesc = GameObject.Find("DropDownMenuDesc").GetComponent<Text>();

foreach (var DifficultyLevel in gameDifficultyHolder)

{

gameDifficultyDropDownMenu.options.Add(new Dropdown.OptionData() { text = DifficultyLevel.Key });

}

gameDifficultyDropDownMenu.onValueChanged.AddListener(delegate { SelectedItem(); });

}

/\*\*

\* Checks for selected game difficulty

\* and assign right customer numbers and times for it

\*

\*/

private void SelectedItem()

{

int index = gameDifficultyDropDownMenu.value;

string selectedOption = gameDifficultyDropDownMenu.options[index].text;

foreach (var DifficultyLevel in gameDifficultyHolder)

{

if (DifficultyLevel.Key == selectedOption)

{

gameDifficultyDesc.text = DifficultyLevel.Value;

}

}

GameController.GameInstance.gameDifficulty = selectedOption;

if (GameController.GameInstance.gameDifficulty == "Easy")

{

GameController.GameInstance.gameTime = 240;

GameController.GameInstance.numberOfCustomers = 8;

}

else if (GameController.GameInstance.gameDifficulty == "Medium")

{

GameController.GameInstance.gameTime = 210;

GameController.GameInstance.numberOfCustomers = 10;

}

else if (GameController.GameInstance.gameDifficulty == "Hard")

{

GameController.GameInstance.gameTime = 180;

GameController.GameInstance.numberOfCustomers = 12;

}

}

// Update is called once per frame

void Update()

{

}

/\*\*

\* Initialising input fields

\*/

private void initiliseCharacterNameInputField()

{

characterName = GameObject.Find("CharacterName").GetComponent<InputField>();

characterName.onValueChanged.AddListener(delegate { setName(); });

}

/\*\*

\* Setting game data character name

\*/

private void setName()

{

GameController.GameInstance.characterName = characterName.text;

}

/\*\*

\* Initialises buttons on the screen

\*/

private void initiliseButtons()

{

backBtn = GameObject.Find("BackBtn").GetComponent<Button>();

backBtn.onClick.AddListener(delegate { LoadSceneByNumber(0); });

quitBtn = GameObject.Find("QuitBtn").GetComponent<Button>();

quitBtn.onClick.AddListener(delegate { exitFromtheEditor(); });

makeCharacterBtn = GameObject.Find("MakeCharacter").GetComponent<Button>();

makeCharacterBtn.onClick.AddListener(delegate {

submitCharacter();

LoadSceneByNumber(3);

});

}

/\*\*

\* Creating the character and setting created to true

\*/

private void submitCharacter()

{

setName();

GameController.GameInstance.itemList.Add(new InventoryItem("Burger", 12));

GameController.GameInstance.itemList.Add(new InventoryItem("HotDog", 12));

GameController.GameInstance.created = true;

}

/\*\*

\* It loads different scenes

\*/

private void LoadSceneByNumber(int sceneNumber)

{

SceneManager.LoadScene(sceneNumber);

}

/\*

\* Handles exit button functionality

\*/

private void exitFromtheEditor()

{

// save any game data here

#if UNITY\_EDITOR

UnityEditor.EditorApplication.isPlaying = false;

#else

Application.Quit();

Application.Quit();

#endif

}

}

///////////////////////////////////////////////////////////////////////////////////////////////////

LeaderBoardHandeler.cs

public class LeaderBoardHandeler : MonoBehaviour

{

private Text topPlayerEasy;

private Text topPlayerMedium;

private Text topPlayerHard;

private Text topPlayerEasyScore;

private Text topPlayerMediumScpre;

private Text topPlayerHardScore;

private Button back;

// Start is called before the first frame update

void Start()

{

initialiseElements();

}

// Update is called once per frame

void Update()

{

updateScores();

}

/\*\*

\* It initialises the text elements on the leader board screen

\*

\*/

private void initialiseElements()

{

topPlayerEasy = gameObject.transform.Find("EasyTopPlayer").GetComponent<Text>();

topPlayerMedium = gameObject.transform.Find("MediumTopPlayer").GetComponent<Text>();

topPlayerHard = gameObject.transform.Find("HardTopPlayer").GetComponent<Text>();

topPlayerEasyScore = gameObject.transform.Find("EasyTopPlayerScore").GetComponent<Text>();

topPlayerMediumScpre = gameObject.transform.Find("MediumTopPlayerScore").GetComponent<Text>();

topPlayerHardScore = gameObject.transform.Find("HardTopPlayerScore").GetComponent<Text>();

back = gameObject.transform.Find("Back").GetComponent<Button>();

back.onClick.AddListener(delegate { SceneManager.LoadScene("Scene\_Menu"); });

}

/\*\*

\* It updates scene with player high scores

\*/

private void updateScores()

{

topPlayerEasyScore.text = GameController.GameInstance.HighestScore[0].ToString();

if (GameController.GameInstance.HighestScore[0] > 0)

{

topPlayerEasy.text = GameController.GameInstance.topPlayer[0];

}

topPlayerMediumScpre.text = GameController.GameInstance.HighestScore[1].ToString();

if (GameController.GameInstance.HighestScore[1] > 0)

{

topPlayerMedium.text = GameController.GameInstance.topPlayer[1];

}

topPlayerHardScore.text = GameController.GameInstance.HighestScore[2].ToString();

if (GameController.GameInstance.HighestScore[2] > 0)

{

topPlayerHardScore.text = GameController.GameInstance.topPlayer[2];

}

}

}

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

A portion of my code in PlayerController.cs

else if (gameObject.tag == "Player" && collision.gameObject.tag == "Soda") // If there is a collision with this tag the burger will destroy itself instantly

{

bool exists = false;

for (int i = 0; i < GameController.GameInstance.itemList.Count; i++)

{

if (GameController.GameInstance.itemList[i].name == "Speedups")

{

GameController.GameInstance.itemList[i].count++;

exists = true;

}

}

if (!exists)

{

GameController.GameInstance.itemList.Add(new InventoryItem("Speedups", 1));

}

GameController.GameInstance.GainedSpeedUps++;

Destroy(collision.gameObject);

}