Technical Design Document

Contents

Game Details

- **Game Name:** *Office Freakout/ Breaking Greg*
- **Team Name:** *Team 4*

Team Members

Name	Job Title	Responsibilities/Roles
Luke Stanbridge	Programmer	Player Controller Menu and UI System
Blake Page	Programmer	Environmental Damage System Randomized Room System AI System

Game Concept

Isometric “hack & slash” game set in a corporate office environment. The game uses physics-based combat to inflict environmental destruction in the office and high impact violence on co-workers. The player objective is to cause as much destruction as possible for HR to deal with. The game can be referenced as a simple version of “Devil May Cry” meets “Stanley Parable”.

Technical Goals

Technical Goal 1 – Competent Player Controller

Who’s Responsible: *Luke*

Description: Need to implement a competent player controller for the user to engage with. Whether this uses a player state machine or a simpler solution to achieve this, it might be worth trying a few different solutions to see what works best. This will need to control the characters movement and combat actions smoothly and feel good to play. I also want to plan and code this in a way that is structured to a professional standard.

Technical Goal 2 - Enemy AI

Who’s Responsible: *Blake*

Description: The enemy AI will need to react to the player's presence by following or running away from the player. Some AI will hide in certain parts of the level, they will also need to try and call the police at the phones placed in the level. Certain AI will be able to attack and stun the player. Enemies will need to be stunned, die, and be destroyed. To make all this work will require a good AI state machine or a similar solution.

Technical Goal 3 – Destructible / Interactable Objects

Who's Responsible: *Blake*

Description: This system will require the player to destroy objects in the environment, having them shatter and despawn over time, while also allowing the player to pick up objects and interact with them. There are going to be a lot of moving parts with this feature so a competent coding solution will have to be implemented. May require a few iterations to see what works best.

Technical Risks

Technical Risk 1 – Framerate and Stuttering

What's the risk about: Potential for high processing power of CPU if the team decides to include lots of destructible or interactable objects in level. If we do have a lot of these objects, processing power can be inundated by repetitive create and destroy calls.

How will risk be mitigated: A good object pooling system to optimise the project and lower the burden on the CPU. A good solution may require some research. Monitor PC performance as we add more detail to the game.

Technical Risk 2 – Physical Glitches

What's the risk about: Player not interacting with objects correctly, hacking and slashing not colliding with objects or enemies correctly. Determining forcefulness of attacks so that combat feels fun but doesn't knock everything out of the level.

How will risk be mitigated: Make sure all colliders and rigid bodies are implemented correctly. Start with small simple items while working out the bugs early then expand to fill out the level.

Technical Risk 3 – Camera Clipping and Position

What's the risk about: Camera not behaving as desired and can't see player behind objects.

How will risk be mitigated: Implement a system to make sure the player can be seen if they go behind any physical objects in the game. Make early decisions on where the camera will be placed to follow the player and how the level will be laid out so it doesn't interfere with user visibility and functionality.

Technical Risk 4 – Enemy AI

What's the risk about: Making an effective AI system for the player to deal with. Could prove tricky depending how much the group wants to NPC's to influence the player.

How will risk be mitigated: Start with basic AI like movement, following player, spawning and despawning. If these prove to be simple then we can look into extra functionality to allow for smarter AI, in turn improving the standard game loop.

Features/Mechanics/Tasks

Feature/Mechanic	Who's responsible	Scheduled Date
Player - 360 directional runs - Idle - Stunned	Luke	Alpha, 20/11/2022
Combat - Light attack (Punch) - Heavy attack (AOE slam) - Chaos state - Throw objects	Luke	Alpha, 22/11/2022
UI - Score system (Damage to the company). - Timer - Rage System - Combo System (Score modifier)	Luke	Alpha, 25/11/2022
Environment - Destruction and movement of game objects. - Randomised level/enemy layout	Blake	Alpha, 27/11/2022
Enemy AI - Movement - Follow Player - Attack Player - Die - Stunned	Blake	Alpha, 27/11/2022
Menu's - Main menu - Pause Penu - Settings - Instructions - Controls	Luke	Alpha, 10/12/2022

Deliverables

Deliverable	Who's Responsible	Who's the Owner
Executable for PC and Gamepad compatible	Luke and Blake	Client
Source Code and Prototypes	Luke and Blake	Client
Progress Reports	Luke and Blake	Programmers
Testing Reports	Luke and Blake	Programmers
Technical Design Document	Luke and Blake	Programmers

System Requirements

Target Device 1 - PC

Recommended Hardware:

- *CPU: Intel 2.77 GHz Quad Core or equivalent*
- *RAM: 2 GB*
- *OS: Windows XP, Vista, 7*
- *VIDEO CARD: OpenGL 2.1 compatible dedicated graphics card (GeForce 6600/Radeon 9500)*
- *SOUND CARD: Yes*
- *FREE DISK SPACE: 2 GB*

Platform Specific Requirements:

- *Keyboard*
- *Mouse*
- *Monitor*
- *Speakers or Headphones*
- *https://docs.unity3d.com/2020.1/Documentation/Manual/system-requirements.html#desktop*

Target Device 2 - Xbox/Sony/Switch

Recommended Hardware:

- *Xbox One*
- *Xbox Series S & X*
- *PS4*
- *PS5*
- *Nintendo Switch*

Platform Specific Requirements:

- *Console*
- *Controller/Gamepad Functionality*
- *https://docs.unity3d.com/2020.1/Documentation/Manual/system-requirements.html#console*

Third Party Tools

What third-party tools are you using? List Unity with version number and any other tools you might need. Include any asset packs you plan to use from the asset store.

- Unity 2020.3.5f1
- Microsoft Visual Studio 2022
- Lean Tween

File Formats

3D Models - FBX

Textures - TGA

Sound and Audio - WAV

Coding Conventions

- Use pascal casing ("PascalCasing") when naming a class, record, or struct.
- Use camel casing ("camelCasing") when naming a method or variables.
- Use all (“UPPERCASE_CHARACTERS”) for constants.
- Standard C# layout conventions with indents and spacing.
- Standard C# commenting conventions
- Null checks when required.
- Avoid underscores in identifiers.
- Vertically align curly brackets.
- Declare all member variables at the top of a class, with static variables at the very top.

Source Control

Source Control Repository: GitHub

Source Control Client Tools: SourceTree/Visual Studio/GitHub Desktop

Source Control Remote Repo URL: https://github.com/LukeStanbridge/BreakingGreg.git

Ignore/Config file: https://github.com/github/gitignore/blob/main/Unity.gitignore

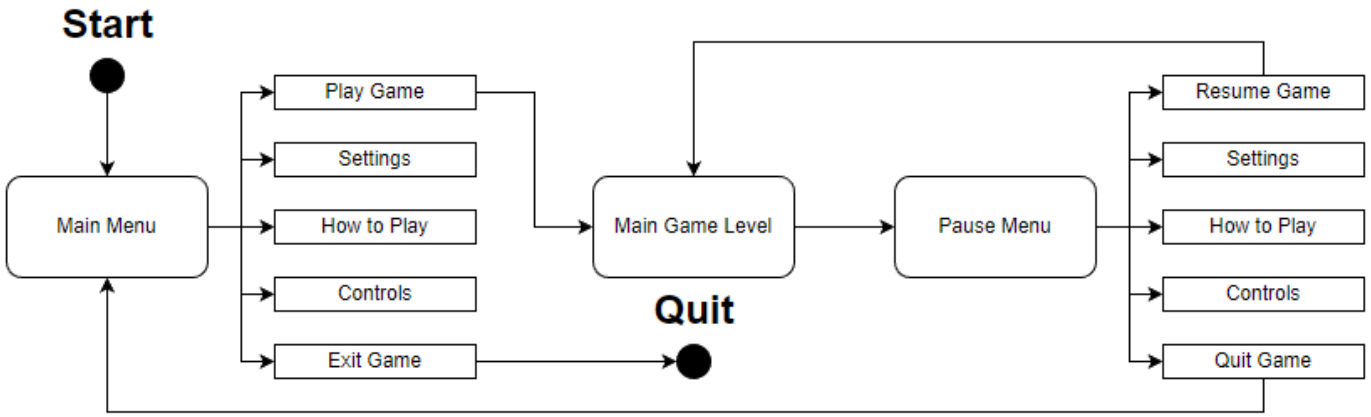
Commit message formats: Description, name and profession e.g. “Fixed Player Controller Bugs - Luke Stanbridge - Programming”

Other repo notes:

- Always pull before you push.
- Don’t change other developers' work without talking to them first.
- Work in separate scenes if required.
- Artists hand off to lead artists before pushing to projects.
- Use separate branches if required.

Game Flow

Scene	Who's responsible	What is does
Main Menu	Luke	Starting menu gives players options to start a new game, game edit settings, game instructions, game controls or quit the game.
Pause Menu	Luke	Pauses the game and allows users the options to resume the game, restart the game, adjust settings, check how to play and what the game controls are and exit back to the main menu.
Main Game Level	Blake and Luke	Load the main game level and manage the main gamestate. Can be interrupted by the player selecting the "Pause" option to open the pause menu.

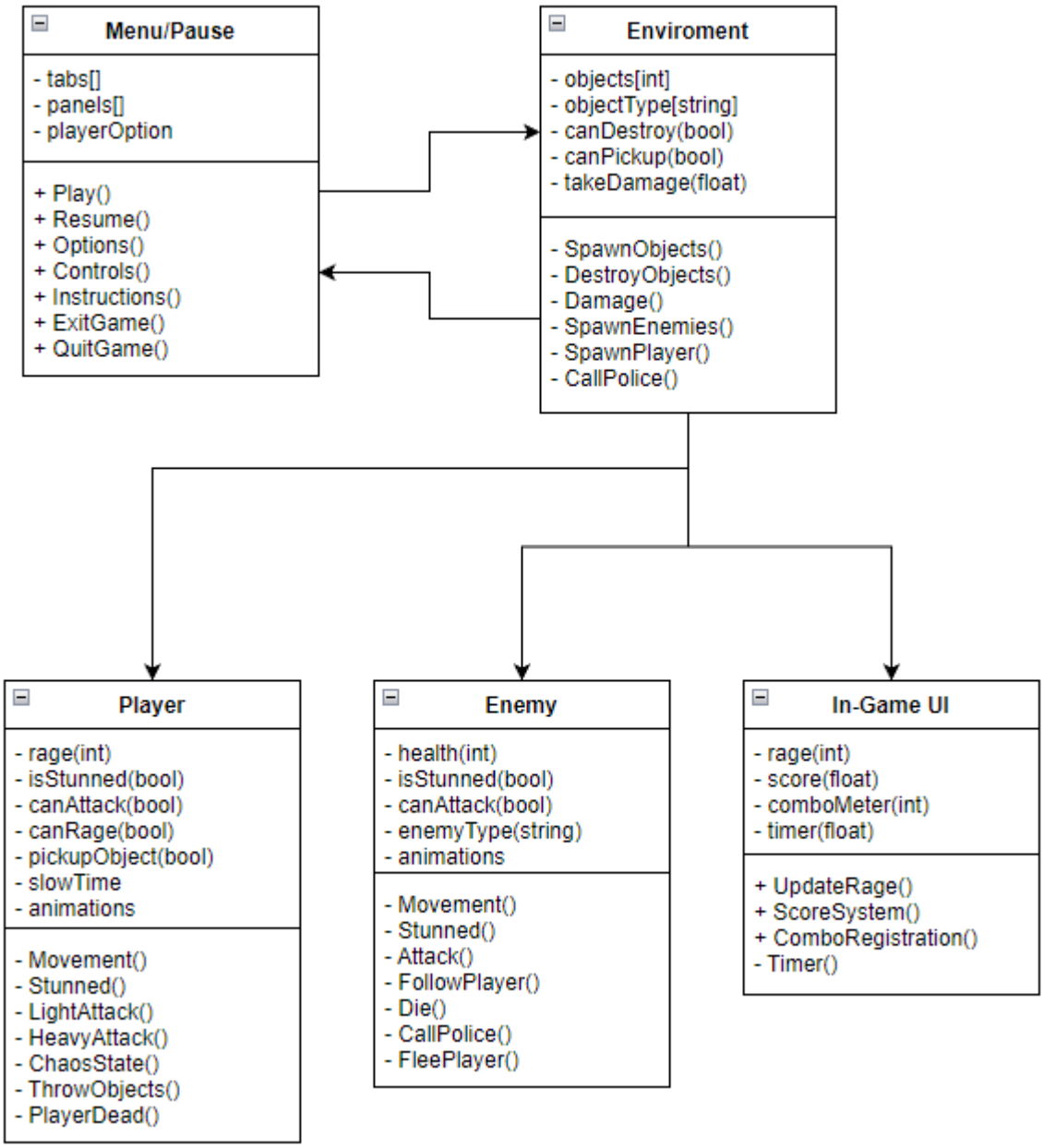


Input Method(s)

Target Platform	Input System	Who is responsible
PC	Mouse/Keyboard	Blake
PC	Xbox360 Controller	Blake
Xbox One/ Series S & X	Xbox Controller	Luke
Nintendo Switch	Switch Controller	Luke
PS4/PS5	Playstation 4/5 Controller	Luke

Game Objects and Scripts

A rough, simple outline of what classes, methods and programming details will be required to set this project up. This will be used as a base to begin with but I am predicting that Blake and I will need to change and go into more detail with this as the project continues.



Gameplay Systems

Gameplay System 1 – Level Structure

Who's Responsible: *Blake*

Description: Each level will contain X number of rooms, these rooms will be prebuilt prefabs that will randomly get assigned to the level for the designers to adjust and fill out the area. An object destruction system will need to be built to handle the office environment getting destroyed. Enemy AI will need to be implemented to attack/follow and die.

Diagrams: *See "Game Objects and Scripts" section.*

Gameplay System 2 – Player Combat

Who's Responsible: *Luke*

Description: The player will have the ability to perform a light attack, heavy attack and the ability to pick up objects in the level and throw them.

The *light attack* will be tied to a button or mouse click. The player will swing/hit whenever the button is pressed and can be used as much as the player desires. The melee attack will inflict damage on any object or NPC and contribute to the combo and rage systems.

The *heavy attack*, which is an AOE slam, can be used to inflict damage all around the player. The attack will have a cooldown timer between uses and add more value to the rage and combo system. This attack will stun larger enemies to open them up to take damage.

The *throwable object system* will allow the player to pick up different medium sized objects and throw them in a forward direction to hit people or the environment. The objects will break on impact and cause damage to the things it hits.

The *chaos mode* will be activatable when the rage meter is full, the player will move at normal speed but the world around the player slows down. Chaos mode will be linked to a timer, once chaos ends the world time will go back to normal and the rage meter goes to half.

Diagrams: *See "Game Objects and Scripts" section.*

Gameplay System 3 – Rage, Combo, and Damage System

Who's Responsible: *Luke*

Description: All player combat that causes damage to NPC's or the environment will be linked to a rage meter system and a combo system to accumulate destruction points which adds to the overall score.

Rage runs down over time during gameplay but goes up when the player kills NPCs. When rage is empty the player loses. Once the player rage meter is full the player will enter the chaos state. Once chaos runs out then rage is set to half.

The *combo system* will tally the amount of consecutive hits the player has before they get stunned or don't kill someone for a couple of seconds. This will simply act as a score modifier to increase the number of points you get per kill.

The *damage system* is just a scoreboard that displays how many points the player has accumulated in the game. Anytime an enemy or object is destroyed the points are added to the total score.

Diagrams: *See "Game Objects and Scripts" section.*

User Interface

Game, Main Menu and Pause Menu UI Layouts.