Technical Design Document
Dos

BRODIE FRAZIER

Change Log:

Version	**Date of change**	**Description**
1.0	29/5/2023	Document creation.
1.1	7/6/2023	Loops updated; diagrams added.
1.2	15/6/2023	Scope update, systems revision and examples of commit added.

Contents:

Development Environment

Used software:

Software	Version	License	Used by	Used for
Visual Studio	2022	Education	Programmer, designer	Scripts
Maya	2023.3	Education	Designers, artists	Models
Photoshop	23.2	Education	Designers, artists	Textures
Source Tree	4.2.0	Free	Designers, artists, programmers	Source control
GIT hub	3.8.3	Free	Designers, artists, programmers	Source control
Unity	2021.3.13f1	Personal	Designers, artists, programmers	Game production and prototyping

Libraries

Name	License	Used for
UnityEngine	Free	All scripts
TMPro	Free	Menu Script
Visual Scripting	Free	Player Cam Script
Unity Visual Scripting	Free	Character Switch Script
Unity Engine UI	Free	Character Switch Script
Unity Engine Scene Management	Free	End Trigger script Menu script

Scripting language will be C# as its compatibility with unity as well as its simplicity and vast capabilities.

Version control:

https://github.com/brazafraza/PrototpyingTest.git

Contributors:

- Brodie Frazier

Commit format:

- Heading (What change was made to, controls, mechanics, level etc…): Description of change.
- Example: Character Controller Script: Added a sprint function to controller.

Game Overview

Genre:

Puzzle, 3D Platformer

Perspective:

First Person

Target Audience:

Indie game / platformer players mainly, all genders, ages 10+ as it involves some puzzle solving capabilities.

Target Platform:

PC / Windows (Steam)

Description:

Dos is a 3D first person platformer single player puzzle game where the player plays as two characters. Each character has a similar but unique character controller which the player will utilize to progress through each level.

The first character (**Harvard)** will be a large character who is able to move objects around by pushing them, with slower and less movement.

The second character (**Beanbag**) will be a smaller character who is able to climb and jump onto objects as well as hit buttons or levers.

The gameplay will consist of the first character moving platforms around for the second character, so they are able to reach the ending of the level.

While doing this, they must collect different collectables as well as being under pressure from a timer, which if hits 0 the level will end.

Feature list:

Character / Player swapping
character controller that can be manipulated
collectables

Game flow & structure

Game mode & Objectives:

Story mode:

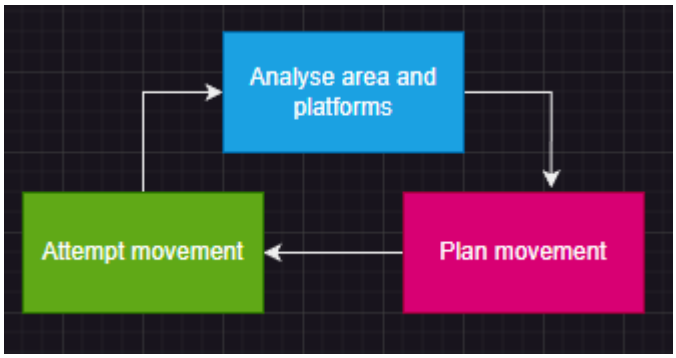
- Player must be able to re arrange the pre generated level to progress to the end.
- Throughout the level, different collectables/trophies can be collected by the player which will be placed in their path of the main objective, as well as out of the way.
- Once the end is reached, they will be brought to a menu were shown which items were collected in the level and their time taken.

Level structure:

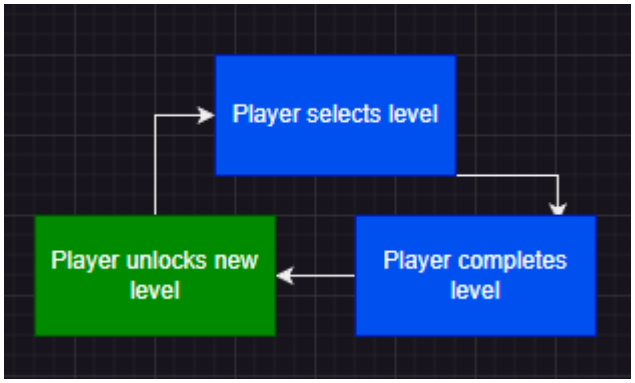
- Player spawns
- Player will identify the end of level / area they must reach.
- Player will create a visual blueprint / plan to reach the ending.
- Player will switch character into Harvard to move the platforms into position.
- Player will switch to Beanbag to attempt platforming.
- If successful, reach end, if not player will identify where they went wrong and try again until successful.
- Once successful, reach end of level.

Game loops:

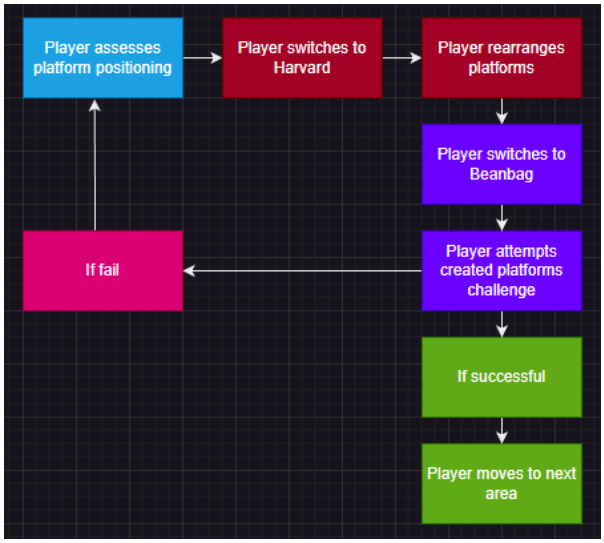
Core Loop



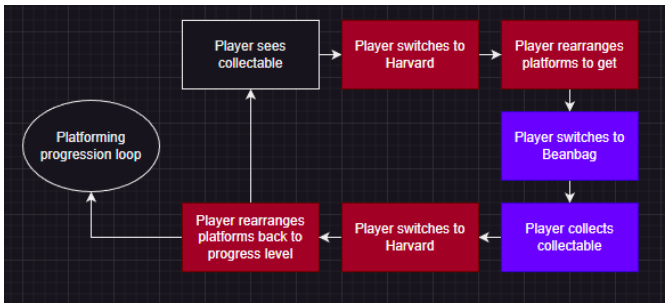
Level Loop



Platforming Progression Loop



Collectibles Loop



Gameplay systems

Mechanics:

- Directional movement
- Cam following mouse.
- Jumping
- Character switching.

Harvard: Will be able to push objects around by walking into them.
Beanbag: Will be 160% quicker than Harvard, able to run faster, able to jump further and higher. As well as being able to use switches and collect objects (collectables).

Similar systems:

- The character controller script used for both playable characters will be the same script however each will be tweaked according to each character's abilities. Harvard will have almost no jump force and a higher mass to push around heavy objects, while beanbag will have the opposite as well as a higher speed value.

Controls:

Standard WASD movement, as well as jumping by space. Character switching by pressing 1 (Harvard) or 2 (Beanbag). Cam follows mouse.

Physics:

- Will be using rigid bodies physics in unity engine.

For example, Harvard will be able to collide with some objects to move them, depending on their mass compared to his.

Prototyping:

Character controllers will be created by a script applying force to rigid bodies to move them within the unity physics system by using keycodes.
Character switcher will be created by a script disabling the game object (and therefore the camera and controller script) for one character at a keycode press and working vice versa to re enable / switch character.

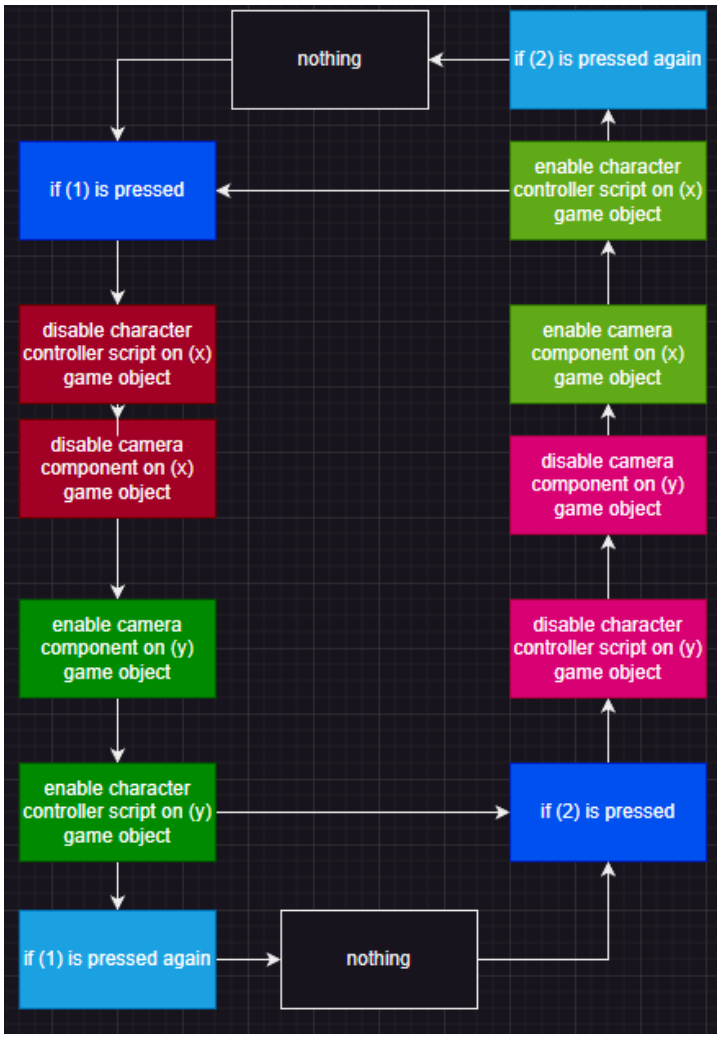
Character Switching pseudocode:

If 1 is pressed, then

1. disable the character controller script and the camera component attached to the Beanbag game object.
2. Enable the character controller script and the camera component attached to the Harvard game object.

If 2 is pressed, then

1. disable the character controller script and the camera component attached to the Harvard game object.
2. Enable the character controller script and the camera component attached to the Beanbag game object.



Character controller pseudocode:

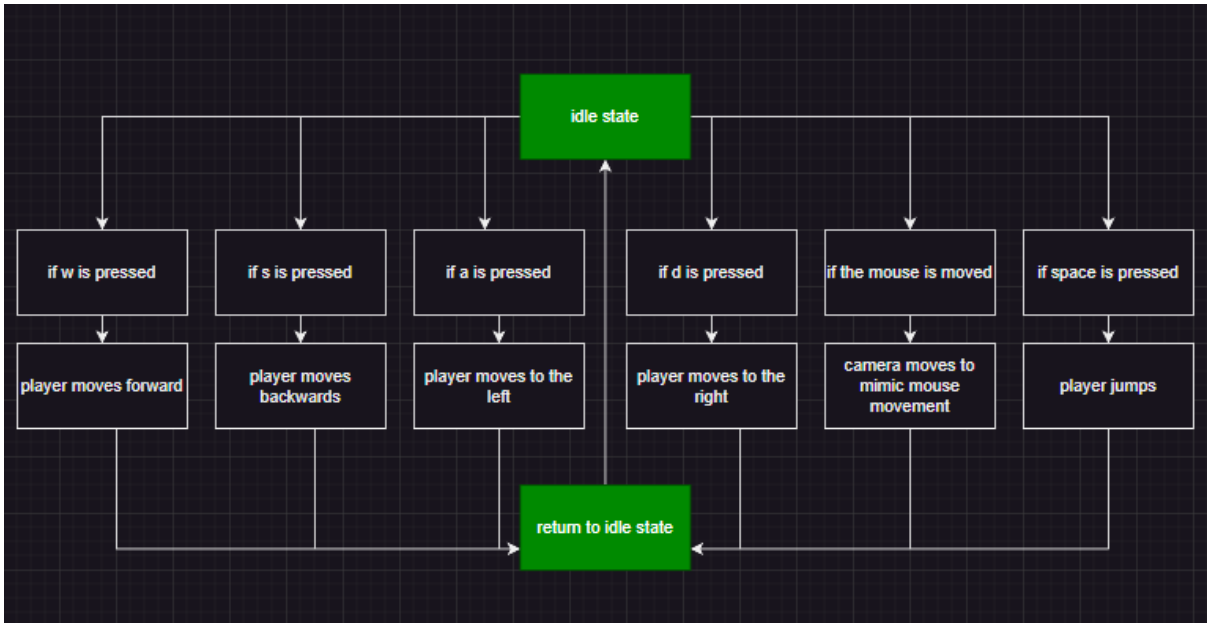
Create a variable for each movement speed, drag, jump force, jump cooldown, air multiplier, if a player is ready to jump

Create a variable for the player height, a mask to place on the ground object and whether or not the player is grounded

Create a variable for orientation aswell an vertical and horizontal input.

Draw a raycast to the floor from the player's centre that is half the players height
If the raycast hits the floor, the player is grounded.

If space is pressed and the player is grounded,
    Apply x force to the player upwards.
If w is pressed, then
    Apply x force to the player forward.
If s is pressed
    Apply x force to the player backwards.
If a is pressed, then
    Apply x force to the player left.
If d is pressed, then
    Apply x force to the player right.
If the player moves the mouse on the X axis, then
    The camera will rotate to mimic mouse movement.
If the player moves the mouse on the Y axis, then
    The camera will rotate to mimic mouse movement.
If the player moves the mouse on the Z axis, then
    The camera will rotate to mimic mouse movement.



Game Content

Levels:

Game will consist of multiple, 3D environments and separate levels. Game will include 12 levels.

Collectables:

Players will be able to collect collectables / trophies throughout the game which have no in game purpose other than being collected. These are recorded and able to be viewed on the menu screen and when finishing a level.

Assets:

Models, textures, scripts, audio files.

Naming & Programming Standards

File naming

Assets:

AssetType_FileName_AuthorFirstName_AuthorLastName_Version

Examples:
Model_Table_Brodie_Frazier_v3
Texture_Table_Brodie_Frazier_v1
Script_CharacterController_Brodie_Frazier_v2

Variable naming

Best precise shortened description of variable, camelCase.

Examples:
type variableName = value;

bool isGrounded = true;
string playerName = “bob”

Commenting

Each comment will be separate by empty lines. Will describe the code directly under the comment.

Example:

// Example comment, explaining the if statement

If
{
}

// Example comment, explaining the else statement.

Else
{
}

Technical goals & risks

Goals:

- Run on solid 60 fps.
- Create smooth UI.

Risks

- Skill gap in certain aspects of the programming language
- Over scope in UI / menu mechanics?

Responses & Contingencies:

- Make sure programmers can successfully prototype these features before the games production.