

An Apple A Day

Technical Design Document

Brooke Feely

VERSION: V.1

***Change Log***

**01-06** V.0 - Template creation. Initial basic (known) concepts included.

**07-06** V.05 - New: Game Overview; Game Flow/Structure; Coding Standards.

**09-06** V.05 - Updates: Libraries list, Mechanics.

**11-06** V.1 - All sections updated to expected standard - ready for project use.

**Contents**

[**Background 3**](#_w8v3pebkspt7)

[**Development Environment 3**](#_pk72dd3eru2k)

[Software Requirements 3](#_332tdbmvbsd1)

[Libraries 4](#_zscg1kw187l)

[Version Control 4](#_4u10dgvf5z2)

[Repository 4](#_c3ecmi9ja6cb)

[Contributors 4](#_ycj6spetvwg7)

[Commit Message Format 4](#_o0mx51joeh63)

[**Game Overview 7**](#_z50ahkdlmc50)

[Description 7](#_24d7ds38epl2)

[Genre 7](#_b1izv8yiic3d)

[Perspective 7](#_9wyk3hbhf3d7)

[Target Platforms 7](#_n31r6v2n4ykv)

[Target Audience 7](#_j68z3c7afcr3)

[Platform - PC 7](#_5ykd783er5ab)

[Min. Req Specs 7](#_6110ok9b57aj)

[Features 7](#_o7afa1s2329m)

[**Game Flow / Structure 8**](#_p9980gkr0t3t)

[Game Modes 8](#_1c84ean6t5cd)

[Game Mode: “Core” 9](#_ihob0kyscz44)

[Description 9](#_rtrsfw5aevfq)

[Objectives 9](#_aa6b87u2bm8r)

[Objective Tracking 9](#_wqzvnt5dkqxp)

[Game Mode: “Panic” 10](#_1skrmzd3rq5y)

[Description 10](#_nh2x608qci00)

[Objectives 10](#_pp9hz1hr9otk)

[Objective Tracking 10](#_o9cnvt7xy1kz)

[Game Mode: “Menu” 10](#_fh6xbrwhss0r)

[Description 10](#_fn4md3pj86jl)

[Objectives 10](#_v4sjt62fps4f)

[Level Structure 11](#_ypxykk5dxi95)

[Overview 11](#_nbbyqmcjzep1)

[Gameplay Loops 11](#_vysz4gr9396a)

[Core Loop 11](#_uakdgmexp3uo)

[**Gameplay Systems 12**](#_e479y7axg47y)

[Controls / Input 12](#_depzbqsds9yl)

[Mechanics 12](#_inqw3mx5e1h)

[Physics 14](#_f8au8g2lbae0)

[Gravity 14](#_zdezmz8ct7i)

[Speed 14](#_jee0ik8ezux2)

[AI 14](#_yz2gkvajqrbj)

[System Comparisons 16](#_nlftixd5usdv)

[**Coding Standards 17**](#_671uhlau2pa)

[Details 17](#_7dai8ehqa8o7)

[Naming Conventions 19](#_9yucetcoell)

[**Technical Goals / Risks 20**](#_bgxn7pd0nksd)

[Goals 20](#_ocbvwlfv4e0x)

[Risks 20](#_6ttyjdvkfutl)

[Solutions 21](#_oh2jt6bqed94)

[**Appendix A 21**](#_s6pfk6bwvhkq)

[Justifications 21](#_kghsdktnr5tl)

[Development Engine 21](#_owocb9uwy6eh)

[Scripting Language 21](#_pu3435jtk3rq)

[3rd Party Libraries 22](#_glr2nea1rjaq)

# 

# Development Environment

Software for use during project creation - contributors are required to use only applications, structures and programs outlined below.

## Software Requirements

| **Software/Program** | **Version** | **Licence** | **Used For** |
| --- | --- | --- | --- |
| Unity | 2021.3.13f1 | AIE Educational | Game Engine / Prototype Creation |
| Visual Studio | 2022 | AIE Educational | Code Scripting |

## Libraries

List of assets, libraries or packages, their validity period and uses within the project:

| **Library (L) / Package (P)** | **Licence** | **Used For** |
| --- | --- | --- |
| UnityEngine (L) | AIE Educational | General Unity functions in code |
| UnityEngine.UI (L) | AIE Educational | User interface functions between code/Unity |
| TMPro (L) | [Unity Companion License](https://docs.unity3d.com/Packages/com.unity.textmeshpro@1.5/license/LICENSE.html) | Text Mesh Pro (text) in Unity |
| Input Manager (P) | [Unity Companion License](https://docs.unity3d.com/Packages/com.unity.textmeshpro@1.5/license/LICENSE.html) | Control Input ease of use |
| Probuilder (P) | [Unity Companion License](https://docs.unity3d.com/Packages/com.unity.probuilder@4.3/license/LICENSE.html) | Physical level prototyping in Unity |

## Version Control

Services used to store, maintain and protect the data created for the project:

### Repository

**<** [Github link](https://github.com/s221247/AnAppleADay_Project.git) >  
(Backed up using SourceTree)

### Contributors

Brooke Feely

User: s221247

### Commit Message Format

All commit messages to be written in English; Summary to be under 30 words.   
Description/Reason kept under 100 words.

**Full Format:**  
Scope (Change Type) : TaskID : Summary.

Description.

**EXAMPLE**UI (FIX) : 07A : Timer HUD text position moved from top-right to top-centre.

Was covering an important area - top-centre had the least noticeable interference to the player view. Other centred placement positions around screen edge were tested.

***Format Explanation***  
  
**List of Scope Areas**

To be used in the full format as above, where “Scope” would be replaced with an abbreviation from the table below, relevant to section being adjusted:

| **Scope Name** | **Abbreviation Used** | **Purpose** | **Example** |
| --- | --- | --- | --- |
| **GAME ART** | **ART** | Animations, 3D models, sprites, shaders etc | When updating a 3D model after second pass. |
| **DOCUMENTATION** | **DOC** | Manuals, documents, excel sheets, PDFs etc as used by (and between) the different departments. | Crucial information update added to the GDD. |
| **LEVEL DESIGN** | **LVL** | Physical design creations in Unity, such as new areas, adjustments, prototypes etc. | When creating a test level layout in Unity for prototyping. |
| **USER INTERFACE** | **UI** | Separate to Game Art and Level Design, purely focused on functionality and placement of features that visually guide the player. Eg. Counts, maps, menus. | Placeholder menu added to game prototype - relevant to Art, Design and Programming departments. |
| **SCRIPT** | **SCR** | All code scripts used within game. | Committing the refactor of a player movement script. |

**List of Change Types**

To be used in the full format as above, where “Change Type” would be replaced with an abbreviation from the table below based on the relevant purpose:

| **Change Type** | **Abbreviation Used** | **Purpose** | **Example** |
| --- | --- | --- | --- |
| **ADDITION** | ADD | New feature, build or script that was not previously present. | New building in the level area. |
| **SUBTRACT** | SUB | Previously added feature/object/etc removed. | Removed a script section that is now redundant. |
| **UPDATE** | UP | Previously added item/object/etc adjusted/updated but kept. Should *no*t be an issue fix. | Refactored a script function to be simpler and clearer to read. |
| **ERROR** | ERR | A bug or issue is present in this version. Should not normally be committed - is for exceptional circumstances. | Version was working in Unity but now is causing serious compiling errors - is happening enough without clear reason to need to show an expert. |
| **ERROR FIX** | FIX | An issue has been fixed, cleared or removed. This should be the usual committed version. | Bug where player was getting stuck in a wall is fixed. |
| **TEST** | TEST | Version is just for experimentation - should be kept separate to prevent loss of current main data. | Version testing whether ragdoll physics would suit the game. |

**Summary**

Under 30 words (ideally).  
Short description of change for clarity of purpose.  
  
**Description/Reason**  
Summary to be followed by a more in-depth description; must include identifying detail eg. lines “45 - 72” of code adjusted, before/after of file names if they were altered, etc

# Game Overview

General outline of the project’s basic descriptors, to get a sense of the game we’re creating.

## Description

‘An Apple A Day’ is a top-down survival action game where you, the protagonist, must escape from a dark, dimly lit dungeon full of DOCTORS. Collect apples to improve visibility, eat apples to regain health or throw them to ward off these grim practitioners to escape this morbid predicament with your life

## Genre

Action / Survival

## Target Platforms

PC / Steam

## Perspective

3D & Top-Down

## Target Audience

PC gamers - all ages, with a focus on mid-teens to late 20s. Fans of survival and survival-horror games, particularly those who enjoy games such as Devil Daggers and Vampire Survivors.

## Platform - PC

PC (Windows/Mac), initial test release through Itch.IO, then Steam.

### Min. Req Specs

Windows:

* + Requires a 64-bit processor and operating system
  + **OS:** Windows 10 64bit
  + **Processor:** Intel Pentium 4 processor or later that's SSE2 capable
  + **Memory:** 1 GB RAM
  + **Storage:** 400 MB available space

MacOS:

* Requires a 64-bit processor and operating system

## Features

* Top-down perspective
* Single player
* 3D game area, objects
* Enemy AI
  + Move independently in random directions
  + Follow player if within range
  + Inflict damage on player
  + “Stunned” by player
* Multiple enemies advancing on player
* Difficulty increase mode near game end
* Lighting
  + Dim play area; lit radius around player
* Inventory & Collectables
  + 1 x collectable affects lighting (increase radius)
  + 1 x collectable usable as both weapon/health
* UI
  + HUD: Player health bar, collectable items count
  + HUD: Message when exit is ready to use
  + Pause screen, menu (start/exit game)
* Enemy AI
  + Wanders the level & chases player when in range
  + Movement paused when hit by a collectable
  + Reaction time affected by game mode
* Sound FX
  + Item pick up, throw and enemy interaction noises
  + Music
* Particle FX
* Level exit (ends game)

# Game Flow / Structure

## Game Modes

**“Menu”** - Static mode; Player chooses to start game or exit.  
**“Core”** - The main gameplay mode: exploration, gathering collectables, enemy evasion, and healing Player.

**“Panic”** - Final mode; difficulty increases in Core mode. Activated when the 3rd apple is collected.

Enemy AI speed increased, cooldown reduced.

## Game Mode: “Core”

### Description

The player starts in the centre of the map, with a minimal radius of bright light surrounding them. They must explore the dark map to find the exit, avoiding the enemy “doctors” as they go.

### Objectives

* Explore the map to find and collect apples
* Evade doctors (using movement inputs to dodge / use apple from inventory)
* Use inventory item ‘red apples’ as deterrents (or use on self to heal if injured)
* Collect all 3 golden apples to increase light radius
* Find exit - door will only open when all 3 apples are found
* Use golden apples on exit door to open and escape (each use will reduce light)

### Objective Tracking

For trackable objectives in “Core” Game Mode:

| **Objective** | **System Used** | **Shown To Player Through** |
| --- | --- | --- |
| Collect apples | Event Management - UI | HUD on-screen during gameplay;  Apple inventory (# count and image sprites) for 2 kinds of Apples |
| Enemy stunned | Event Management - Triggers | Audio-Visual feedback when a successful collision occurs |
| Healing | Input Manager & Event Management - UI | HUD (only active when affected) showing health |
| Exit Ready | Event Management - UI & TMPro | HUD temporary message pop-up;  On collection of 3rd Golden Apple, inform player that they can now head to the exit. |
| Exit Opening | Boolean Code Script  Event Management - UI  Application Class | As player uses Golden Apples on door, light decreases. 3rd Apple removes exit collider, player can exit (once through door, game ends). |

## Game Mode: “Panic”

### Description

The player has collected all 3 golden apples. On pick-up of the final apple, Panic mode is activated. Music gets faster, enemy movement is faster (including reduced cooldowns from red apple hits) and more enemies spawn.

The player must escape and use their golden apples on the door as quickly as possible.

### Objectives

* Find the exit door
* Evade frenzied doctors using movement inputs and throwing red apples
* Use golden apples one by one on door (while defending against enemies)
* Exit door once open to end the game

### Objective Tracking

For trackable objectives in “Panic” Game Mode:

| **Objective** | **System Used** | **Shown To Player Through** |
| --- | --- | --- |
| Exit Ready | Event Management - UI & TMPro | HUD temporary message pop-up;  On collection of 3rd Golden Apple, inform player that they can now head to the exit. |
| Exit Opening | Boolean Code Script  Event Management - UI | As player uses Golden Apples on door, light decreases. 3rd Apple removes exit collider, player can exit (once through door, game ends). |
| Exit Entered | Event Management - UI | Game screen change with end game text  and restart/exit options. |

## Game Mode: “Menu”

### Description

The opening view once the game program is started (or once they have ended gameplay).   
Dark background with simple text UI and accompanied by the image of an apple.

Player can select two options:

### Objectives

* Select Start to begin game
* Select Exit to exit program

## Level Structure

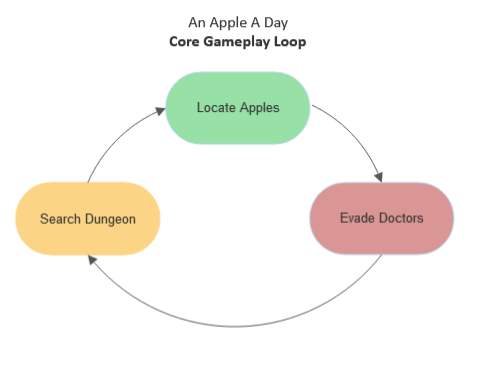
### Overview

The level consists of a dimly lit (near dark) rectangular, maze-like space with walls and doorways to navigate. Player’s starting location is the centre, with collectables littered around the map and various connected (or blocked) pathways to explore and find the items. Exit is located top left of the map. Enemies are littered throughout, more surrounding Golden Apples.

## Gameplay Loops

Repetitive cycles used throughout the game to establish both core and meta experiences - here used as comparisons for script planning:

### Core Loop

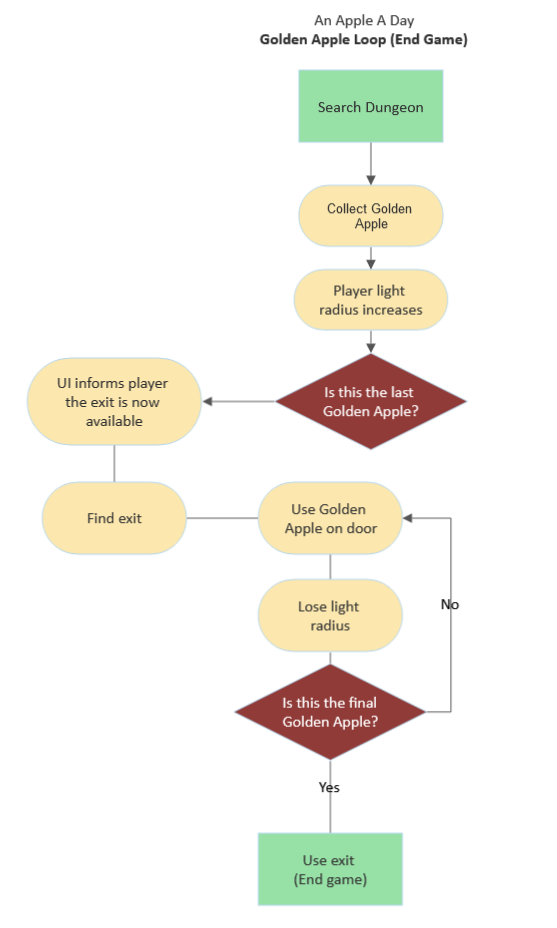
This is the main repeating gameplay cycle - the player must explore, collect and evade until the winning objective (all Golden Apples collected, exit unlocked) is reached.  


***Meta Loops***

Two main meta loops for the gameplay:

* Golden Apple (initial gameplay, end objective)
* Red Apple (combat/heal loop)

Flowcharts for both are on the following pages:



# 

# 

# Gameplay Systems

General outline of the project’s specifications for system structures within the gameplay.

## Controls / Input

Table of game controls - input keys and their function:

| **Input** | **Function** | **Context** |
| --- | --- | --- |
| **W** | Move up / highlight menu option | When pressed OR held down |
| **S** | Move down / highlight menu option | When pressed OR held down |
| **A** | Move left | When pressed OR held down |
| **D** | Move right | When pressed OR held down |
| **E** | Use Golden apple (on exit) | Requires UI prompt at door |
| **Space** | Throw Red apple | Stored in inventory, # shown in UI count, pauses doctor movement when apple collides with them. |
| **Shift** | Use Red apple  (Heal player) | Stored in inventory, # shown in UI count, heals player. |
| **ESC** | Pause menu | In-game option. Halts gameplay when active and shows text UI options. |
| **Enter** | Activate menu option | Press to start/resume or exit game (depending on which option is highlighted) |

## Mechanics

Table of game mechanics - numbers proposed here are examples only, can be adjusted through further testing.

| **Mechanic** | **Purpose** | **Rules** | **Starting Values** |
| --- | --- | --- | --- |
| **Walk** | Move throughout the game area, avoid enemies. | Can walk diagonally. | Speed: 4.5f |
| **Light View** | A circle of light surrounding the player, allowing a brighter field of vision only near them. | Must be circular.  Does not light full screen - centred on player. | Start radius: 10%  Final radius: 40% |
| **Inventory** | Passive storage of collected apples - displayed through UI on the game screen. | HUD inventory view only - no secondary inventory view.  Inventory space has a cap. | Inventory Cap  Golden: 3  Red: 4 |
| **Red Apple** | Used to heal player from Doctor damage or briefly hinder doctors’ movement. | Red Apple hold amount is capped. Thrown or heal are separate inputs. | Red: 4  Throw: Space  Heal: Left Shift |
| **Golden Apple** | Passively used (once picked up) to increase radius of light. Also used to unlock the exit door (decreases light radius). | Cannot be thrown, no cap on maximum amount. | Radius increase: +10% |
| **Collect Apple** | Pick up apples by walking over them; they are added to the inventory. | Red Apples have a capacity limit. | Total Apples In Level  Red: 20  Golden: 3 |
| **Health Bar** | Green UI bar showing state of player’s health. When health reaches 0, game ends. | Health counted in integers. | Max Health: 100 points |
| **Heal** | Use Red Apple to recover health loss to Doctor damage. | Heal counted in integers. | Heal: 25 points |
| **Throw** | Throws a Red Apple which ‘stuns’ a Doctors if they are hit by one. | Direction: Player facing | Distance: 10f |
| **Doctors** | AI enemies - set to chase and injure the player | Random movement around map when not chasing. | Speed: 10  Movement Cooldown: 5 sec  Max # enemies: 15 |
| **Unlock Door** | End of game exit - unlocked by 3 Golden Apples. | Min req. to open: 3 Golden Apples  Player-Door distance min. before interaction: | Min req. to open: 3 Golden Apples  Player-Door distance min. before interaction:  5f |

## Physics

**[ Note:** All values listed as temporary starting point placeholder **]**

### Gravity

Values to be used for weight affecting in-game physics:  
(Determined by use of Rigidbody or Unity Project Settings)

| **Unity Object** | **Rigidbody?** | **Mass Value** |
| --- | --- | --- |
| **Project Settings** | N/A | X: 0f, Y: -9.0f, Z: 0f |
| **Rigidbody: Player** | Y - Dynamic | 1.0f |
| **Rigidbody: Doctors** | Y - Dynamic | 5.0f |
| **Collectable Apple (Both)** | N | N/A |

### **Speed**

Values to be adjusted within Unity inspector:

(Base being the value used in “Core” Game Mode, Secondary used on Doctors in “Panic” Game Mode)

| **Unity Object** | **(Base) Speed Value** | **(Secondary) Speed Value** |
| --- | --- | --- |
| **Player** | 10.0f | N/A |
| **Doctors** | 10.0f | 12.5f |

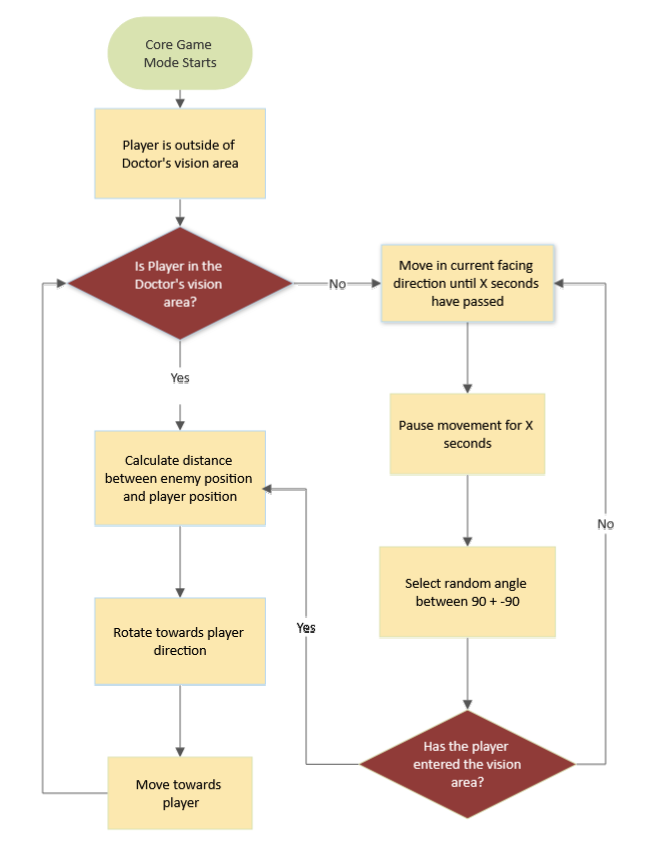
## AI

Enemy “Doctors” are to use scripted random movement/player awareness to follow the player.

Note: If this coding goal becomes too difficult within the time scope, Nav Mesh can be considered as an alternative.

Find pseudocode below as an example of potential coding for the AI:

**Pseudo Code for enemy-player chase system:**



### System Comparisons

**Inventory & Apples**

Two different items (Red & Golden Apples) are located, collected and stored in the same way, their scripts are similar but require important differences. Both use a combination of UI, collider and trigger effects, however:

* Golden Apples are restricted to a certain number due to their use as end goal keys - Red are used more often in combat; they must be more malleable to adjust values with.
* Their functions of use must be kept separate - throwing Golden ones will mess with the lighting system and interfere with game visibility.
* While both apple types have inventory caps, Red ones are reusable - Gold must be kept separate.
* UI and inventory tracking for both are therefore intertwined and should behave the same way - with some exceptions to trigger events for end game notifiers (eg. Last Golden Apple collected).

**Enemy Damage to Player vs Player Stunning Enemy**

The systems for enemy-player interactions are quite different - from movement styles to how they handle collisions.

* Enemies don’t require UI use - they have no “health” and as such scripts should handle a different manner of effect (interrupting movement, not permanent enemy damage).
* Whereas the player has finite health, which should be reduced with every enemy collision and shown in the player’s HUD.
* This “one-sided” approach to combat aims for a sense of challenge and fear-inducing lack of control in the game, so along with randomised but targeted enemy movement, these systems share similar interaction points but not similar structure.

# 

# Coding Standards

Entire project’s coding conventions to be used during development.

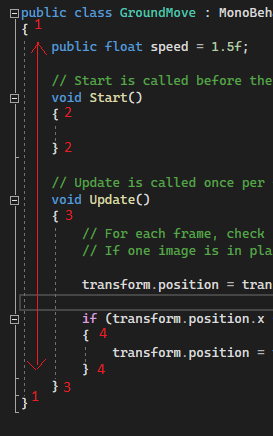
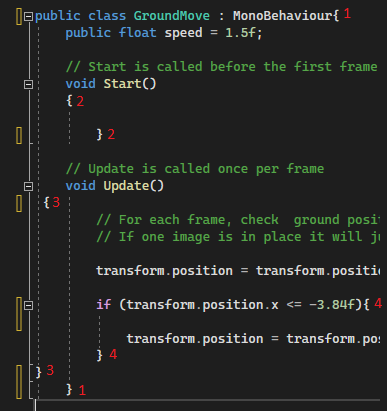
## Details

1. Open/close braces per section (incl. nesting) to be kept in line with itself in each set.

**Examples:**

In A1 (left) below, braces are **aligned**. Individual open/close sets having matching numbers.

Example A1. Correct Alignment. Example A2. Incorrect Alignment

****

Meanwhile in the A2(on the right), the same braces are **not aligned**, meaning they are harder to locate and generally messy.

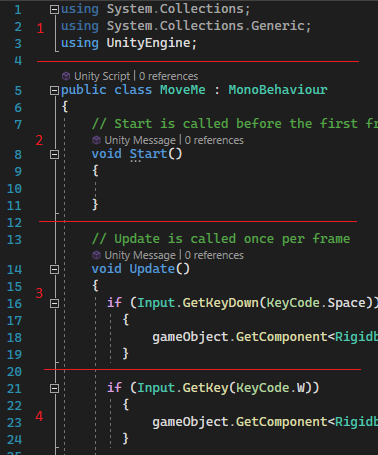
1. Maximum of 1 line of white space **between** each **new** section of code and between new code lines that focus on a different concept/object or function. The code lines at the start of a new section do not need an extra space, but should be on a new line (as shown in the left image below).

Eg. In exampleB below, new sections are distanced with one empty line between.   
  
 1. Libraries are kept separate from the entire code by one space.

2. The overall code scope begins, with the relevant braces one line below. Braces are kept on a new line. Commenting is kept on the line directly above the relevant code it precedes.

3/4. Space between new if statements, even though they’re in the same section - each is a new concept.

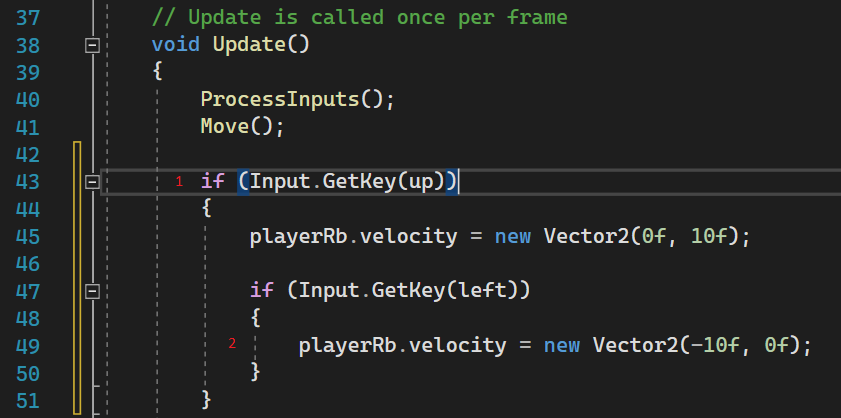
Example B. Clear spacing.



C. Nested code within code sets to be indented to the right with each new set added (**once** per new layer).

Allows connected code to be more easily identified.

Eg. In the (nonsensical) code below, the second if statement is indented across to the right.



## Naming Conventions

Listed below are the conventions for naming files, functions, variables etc.

Coloured text denotes the adjustable part of the name, as names of objects may be different but the identifiers (in black in Format/Example) should remain the same. All names have no spaces between each new word.

| **Context** | **Format** | **Reason** | **Example** |
| --- | --- | --- | --- |
| Script File Name (Class) | PurposeOfScript  (Pascal Case) | The script title should clearly show (without reading the code) what the use is, and what it is for. | EnemyAwareness  (Everything within this script is related to the enemy’s awareness) |
| Function | VerbPurpose  (Pascal Case) | The function name should clearly show (without reading the code) what the use is, and what it is for. | awareOfPlayer  (Function for whether the enemy can “see” the player) |
| Rigidbodies | objectnameRb  (Camel Case) | Clarity of relevance  (the name of the object this rigidbody attached to); shortened to reduce typing. | enemyRb  (Rigidbody component in Unity in this instance is on an object called “enemy”) |

# 

# Technical Goals / Risks

Project’s main aims, difficulties and ways to manage them.

## Goals

* Produce a playable prototype using at least 3 coding scripts.
* Deliver final prototype by due date.
* Have tested/checked final prototype code multiple times before delivery.
* Game length lasts at least 10 minutes.
* Creating a functional inventory that can be reduced/increased after interacting with the environment.   
  (eg throwing apples at AI)
* AI scripting that causes enemies to follow and respond to certain stimuli/the player.  
  (“seeing” the player when in range, following and surrounding the player, dispersing when apples are thrown near)

## Risks

* Beginner coding team requires extra time to learn new scripting methods.
* Project data loss due to damage, power outage, site outage   
  (increasing workload, reducing scope, possibly missing deadline)
* Progress/timeline setback due to saving over important data   
  (reducing scope, increasing workload)
* Missing deadline due to time lost on inefficient coding practice.
* Time lost through extended debugging,

## Solutions

* Continually assess progression and time taken for each task - eg. If the task takes more than 24 hours to complete, consider a new route.
* Opting for simplified code where possible - use the path of least resistance.
* Screening tutorials for authority, relevance and efficiency of content for the coding team.
* Call in outside assistance from AIE team for code/concept clarification.
* Save multiple backups of project data in different areas - External Hard Drive, Source Tree, GitHub, Home Computer, Google Drive.

# 

# 

# Appendix A

## Justifications

Reasoning for chosen development methods within this project with comparisons.

### Development Engine

**Unity** (vers. 2021.3.13f1)

* Selected as we hold the licence for this program;

the project can be worked on both remotely and in-studio.

* Easy to use and relevant to the developer team’s current skill set  
  (they have previous experience with the Unity interface).
* Compared to other available free engines such as Godot, Unity has a much stronger toolkit for 3D graphics - important for this game due to the heavy reliance on lighting and mood. In contrast, Unreal 4 and 5 have stronger 3D effects in comparison to Unity, however their processing power would be unnecessary as this project is quite simple.

### Scripting Language

**C#**

* Chosen as this language is best suited for ease of use in the Unity Engine.
* Optimised compatibility with multiple programs (Unity, Visual Studio) which makes it a strong foundational language to use for smaller projects like this one.
* Current programming team has some experience with C#, but no experience with C++, which has a much steeper learning curve. Time would be lost on learning new code like C++, and for the project’s scope it does not need extensive language.

### 3rd Party Libraries

**Probuilder**

* Designed specifically for use in Unity to streamline level building process.
* Reduces time spent on prototype creation.
* Development team has experience with its use.
* Has been purchased by Unity to be included in their packages - allows for on-going support with any issues. Note: Still considered to be in “experimental” stages.