Technical Design Document

SOVA Training Homework

BeeYou

Fleur's Fabel

Game Mechanics/Features

- All the text and Dialogue in the game will be in Dutch
- Player
  - Third person perspective
  - Walking
  - Collecting by walking over objects
  - Talking with a ‘shockwave’ button
  - 3 Different types of body language
    - passive, aggressive, assertive
    - Player can toggle between each type of body language
  - Part of character changes color depending on current emotion
- Game world, including
  - 3D objects
  - NPC’s
  - Semi-open world
  - Player can pick an emotion at emotion shrine
- 7 Different levels
  - Each level handles a different topic of the SOVA training
- Audio and sound effects
- HUD
- Menus
  - Main menu
    - Settings
    - Audio Settings
    - Credits
  - Pause menu
    - Settings
    - Audio Settings
    - Back to main menu
- Save system
  - on application quit
- Data gathering
  - Name
  - Time played per level
  - Sending data to database through php scripts

Game Engine

What is Unity?

For the development of the game, we will use the game engine Unity3D. Unity is a game development software, which can be used to create 2D and 3D games as well as mobile applications and interactive simulations.

Unity Features

Some but not limited to:

- C# Scripting
- Visual Scripting
- Physics Engine
- Animation
- Particle System
- Multiplayer Networking

Why Unity?

Unity is a simple environment for making games. It has a clear UI and a lot of built- in functions like a physics engine, animation system and a particle system. Unity also has a lot of good documentation and a lot of tutorials, so it's easy to look something up if you don't know how something works or if something broke and you don't know how to fix it.

Another reason for choosing Unity is because they have an asset store where we can find a lot of asset packs and separate assets. So if the artists don't have enough time, or we quickly need assets we can always get them from the asset store.

Unity is also a good choice for us, because everyone in our team has the most experience in working with Unity. If we were to choose a different game engine, the entire team would need to get used to a different game engine and it would slow down production time.

Which version of Unity?

We will be using Unity version 2020.3.21f1 LTS.

We will be using that version, because it is the most recent long time support from Unity. That means that it will be updated frequently and it will have support for multiple years.

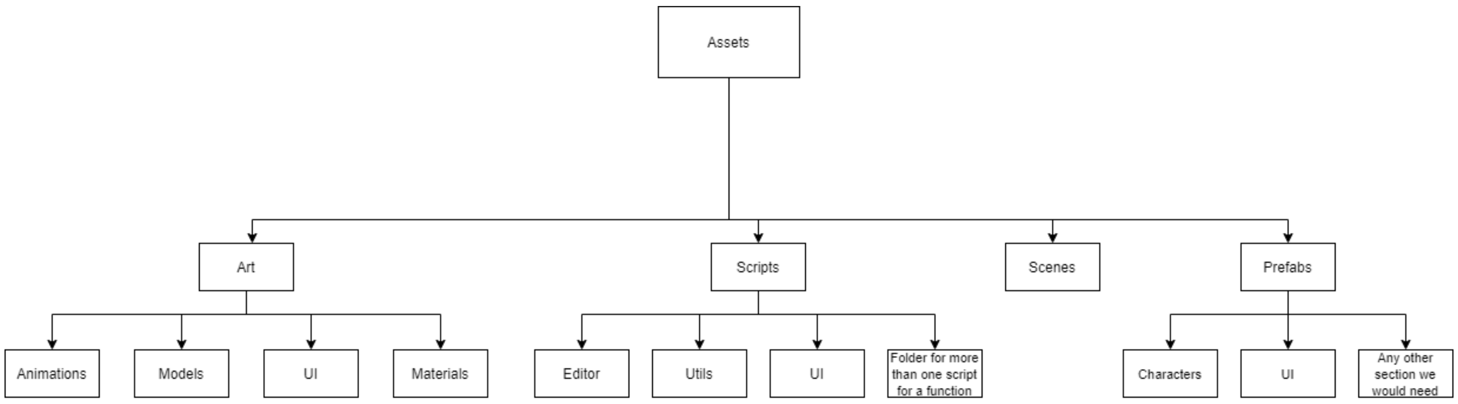
Plan of Action

We use Trello for our project planning. You can click here to view the Trello board.
We also have bi-weekly meetings with the client to give them updates, ask for feedback and ask questions if we have any.

Diagrams

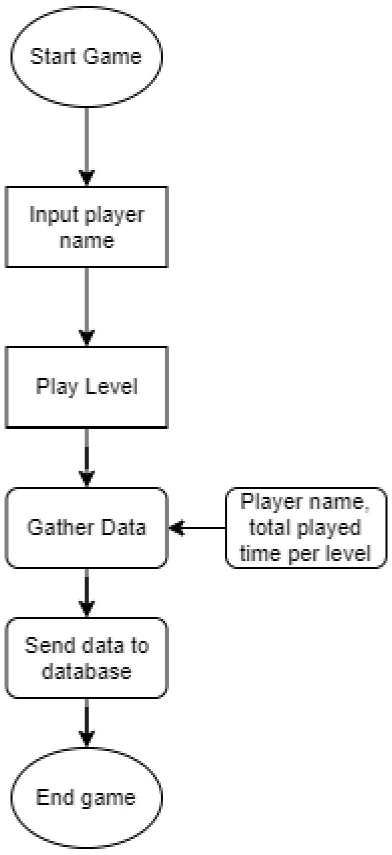
File Hierarchy Diagram

This diagram shows the file hierarchy, so what folders will exist and how they're grouped. This is still open for change, even during the production phase.



Data Diagram

The data diagram explains, in a very basic form, what happens with the data we gather from the player.



Asset List

- Player character
  - Possible multiple version for character customization
- NPC’s
- Environment
- UI
  - Menu’s
  - HUD
- Fonts
- Splash Screen
- Sounds
  - Music
  - Sound Effects
- Visual Effects
  - Particles
- Shaders

Collision, Physics and Interaction

Collision Detection

Collision detection is a check between 2 or more objects if they are intersecting (colliding) or not. Below is a list of places where a collision check is needed.

- Soundwave needs to hit an npc and cannot go through walls
- Player and npc’s collide
- Player cannot walk through objects
- Npc’s cannot walk through objects
- Player can pick up certain items
- Player cannot go through the border of the map

Physics

Physics is what makes the game feel realistic. Physics are things like gravity, how items fall or break, how a character’s speed accelerates and stops etc. For all the physics in the game we will use the Unity physics engine and if needed we will make our own functions for calculations and/or physics mechanics.

Below is a list of physics components that will be in the game.

- Collision detection
- What must be realistic:
    - Items & objects fall realistically
    - Movement of player and npc's
- Gravity
- Friction

Interaction

This is how the characters (player and npc's) interact with the world and each other.

- Player interaction
    - Talk with certain npc's by doing a talking 'shockwave'
        - 3 different types of shockwaves
        - Passive
        - Aggressive
        - Assertive
    - Pickup items
    - Choose emotions at an emotion shrine
- NPC interaction
    - Talk with the player if triggered by the player
        - Different types of reactions based on the type of talking shockwave of the player

Game Logic and AI

Game Logic

- All the game logic will be made with C# and the Unity API
- PHP scripts will be used to send data from the game to the database.

AI

The npc's will use simple AI algorithms for things like walking and choosing a reaction for dialogue, based on the input of the player. Below is a more detailed list of what type of AI we will use.

- Some npc's will be walking around (e.g. in a town)
    - They will walk around in a *(pre-determined?)* pattern
    - If the player talks to an npc it will stop walking and face in the direction of the player
- Npc's will have different types of reactions based on how the player reacts to them.

    - Different types of reactions
    - Passive, aggressive, assertive

Audio & Visual Effects

You can check the Art Bible to see how we are going to make the audio and visual effects. The audio and visual effect lists are still WIP and effects can still be added later.

Audio Effects

- Footsteps
- Dialogue
- Ambient sounds
- background music
- Talking 'shockwave'
    - passive
    - aggressive
    - assertive
- pickups
- emotion change

Visual Effects

- Footstep dust
- Talking 'shockwave'
- Pickup effect
- Mood/emotion switch
- NPC attention thing

Target Platform & Hard/Software Requirements

Target Platform

PC Windows

Hardware Requirements

TBD

Software Requirements

TBD

Coding Standards

Naming Conventions

These naming conventions will be used for the variables in the scripts. It is important that everyone uses the same naming conventions, so that everyone can easily see what type of variable it is. It also makes sure that the scripts will look the same so it's easier to merge them if needed.

Script names

ExampleScript

Script names will not have any special characters, only regular text and numbers.

Global Variables

private _camelCase
public CamelCase

Local Variables

private camelCase

Functions

private camelCase()
public CamelCase()
parameters pCamelCase

Style Guide

All global variables will be at the top first grouped by access modifiers (e.g. private and public) and within the access modifiers they will be grouped by variable type (e.g. int, float and bool) and if they are serialized or not.

Below the variables the functions will start. The standard Unity functions will be at the top (e.g. Awake(), Start() and Update()) below the standard Unity functions will be the private functions and after the private functions will be the public functions. Functions don't have to be ordered by function return type.