A Guide to YCRB5’s Programming Assignment 3



Well, well, what do we have here? If you guessed “an insanely good looking, complex, over-the-top 2D game in the guise of an experiment that took weeks of non-stop work to create and was good practise but also definitely way too much work” then you’re certainly right! This guide talks you through what’s going on, some plusses and some downs.

**What experiment are we doing here?**

Research question: “What choices do players make in video games and are these choices internally consistent?”.   
The experiment records the choices players make in a custom-made video game focussed around interaction and exploration. In exploring the world, the player will interact with multiple NPCs (non-player characters) and is given the option to respond. Three dialog choices are always available: a friendly/polite option, a neutral option, and a rude option. The player’s choices of these dialog options are recorded. These can later be analysed to look at differences in responding between NPCs, average/maximum/minimum responses type, individual differences and order in which the player took part in the dialogs.  
The game starts at the beginning, so players will slowly get acquainted with the world and characters.

**How does it work?**

The experiment runs in two parts:

1. The demographics and information sheet. These are filled out by participants prior to starting. They are collected by running the “Demographics.py” file.
2. Once the demographics have been filled out, the players move on to the game. This is started by running the “Game.py” file. The game begins with some exposition in the form of dialog, after which the player is relatively free to move around the map and explore. Dialogs are triggered by entering buildings. Some buildings have to be entered at different stages in order for the game to progress. After the player has gone through the entirety of the story, the game will quit by itself. The minimum amount of dialog choices a player can make is 31. The maximum amount is 51 choices.

**Wanna play?**

The programs run smoothly by themselves, provided all referenced files are in the right place. It is especially important that the img, dialog, sprites, fonts and tiles folders contain the right assets and that the .tmx, .tsx and .py files are in the main folder. It is recommended to have the pygame library installed as well.

Adding new dialog for an existing location is easy: copy one of the existing .xml files, follow the structure presented there for entering text and player options, and safe the file with the name of the location and the next available number.

Adding a new location and dialog is currently harder to do (although not too hard), and requires going into the .tmx and .py files to add code.

All participant demographic data can be found in the participants/participantsData.csv file. All player choice information can be found in the participants/[player name] choices, [date].csv files. A new player choice file will be created for each player, provided the player doesn’t use the same name as an existing player on the same day.

**And now for something completely different…**

The experiment makes use of some quite complex code to achieve quite complex things. This code is explained in the appropriate files. However, for a quick overview, here is what the program currently can and can’t do. The focus here will be on the Game program and not the Demographics program.

**The Good:**



* Interactable overworld map in the style of Pokemon, but also NPC dialog in a visual novel style, with seamless transitions between the two

**Overworld map**

* Beautiful pixel art graphics
* Collision detection, entry detection
* Depth: player can disappear behind objects, objects can occlude other objects
* Overworld scrolls with the player position
* Illusions of height due to placement of map objects (stairs, rocks)
* Separate tilesets for different parts of the map (created by hand from larger files)
* Custom made player & companion sprites
* Player sprite walking animations
* Collision detection on bottom half of the player only, allows for better control
* Overworld map is extremely easy to modify e.g. w/ Tiled and reimports into program automatically

**Dialog**

* Easy to add new dialog for existing locations
* Large variety of NPCs with their own dialog boxes and locations
* Ability for player to select and confirm dialog choices, allows for responding/interacting with NPCs. Each choice is written on a new line that players can select.
* Easy to write and modify dialog by using .xml files which are automatically loaded based on location
* Backgrounds, NPCs and dialog boxes are automatically loaded based on location
* Text wraps to new line when maximum length of dialog box is reached
* NPCs can respond to the tone of the player but don’t necessarily have to
* Some dialog does not become available until a certain location has been visited
* Dialog changes depending on how often player has visited a location
* Ability for player to enter their name/alias by using their keyboard. This name is then used by NPCs to refer to the player throughout the rest of the game.
* Multiple NPCs can be present and interacting with the player/each other in one location. Which NPC is visible is specified in the .xml file.
* When the dialog has finished, the player automatically exists the location. Some dialog will only be displayed once.
* When the dialog has finished, the player’s choices are written to a file and saved

**The Bad:**



Some things did not go as planned. These are detailed here.

* NPCs can only react to one out of three possible lines for every player choice, and can only respond using 1 line the .xml file. This means no branching paths are possible within one dialog.
* Adding a new location requires adding code and changing the .tmx file. Unfortunately, this is more involved than just adding more dialog to an existing location. (See “The Ugly”)
* Only one depth is possible relative to the player. E.g. if a player disappears behind a roof, they can’t then also climb on top of the roof later
* Only one NPC can be visible on the screen at any one time, even when there are multiple at the location.
* Players need to enter different username when playing on the same day to prevent overwriting a previous file. (Also solved by playing with the same name on a different day).
* The companion sprite does not follow the player sprite, as originally planned. This was too involved and I didn’t have the time.
* The game is shorter than planned.
* Dialog text sometimes already continues at the slightest press of “E”
* Some features would be nice to have, although not necessary. For example:
  + NPC expressions that match what is said
  + A battle screen, based on Undertale’s mechanics with options to Talk or Fight, and attacks/dodges determined by QTEs (Quick Time Events, quick button presses). This was a big part of the original plan, but I ran out of time
  + sounds
  + scrolling text in the dialog screens

**The Ugly**



Some solutions needed to be hardcoded, either because I ran out of time or couldn’t find another way to do it. These are detailed here.

* The locations. This is the main one that really annoys me. Each location needs to be coded into the program, in order to make lists that match the location names and determine which dialog to present when the play enters it. List variables that get created when a player enters a location won’t do, as the Dialog file then doesn’t save whether the player has been there before. This currently isn’t a problem, but will be if the game gets more complex. Haven’t yet been able to find a better solution
* The Start dialog. This dialog isn’t triggered by a location in the world, therefore it wasn’t possible to load it like other dialog files. It also required keyboard input from the user in the sense of their name. This dialog is currently a separate file, run once before the overworld map is even loaded. Afterwards, the map loads and the game starts.
* Screen size. Lots of things are hardcoded or designed according to the size of the screen, e.g. the dialog boxes, sizes of tiles in the overworld map, sprite size, NPC images and more. This means the game window be scaled up/down. The current aspect ratio is 19:6, with a width of 900 px and height of 506.
* Movement/confirmation keys. These are currently hardcoded, although extremely easy to change. E.g. in the beginning stages of development I played with the arrow keys and “/”, before deciding on WASD and “E”.
* Some dialog is hard-coded, especially with respect to location. E.g. if the player hasn’t gone to the dojo yet, the dialog in the house location is set to always be house2.xml. Afterwards, house3 and house4 become available.
* The code that gets the appropriate dialog line after a player choice has been made (i.e. checks if the NPC responds to the player’s tone or just continues as normal) is wonky. Even I don’t really know what’s going on, and I wrote it. However, it works.

**Credit where credit is due**

However highly (or lowly) you may think of me, I didn’t not create all the assets that are used in this program. Nor all the code. Please find a full list of credits down here.

**Tilesets**

Tileset for the snow landscape: separate tilesets created by me from image found here: <https://fanart.pokefans.net/tutorials/mapping/tilesets>.

Tileset for the grass landscape: tiles are from Pokemon FireRed/LeafGreen. Separate tilesets created by me from image found here: <https://www.deviantart.com/ozotwo/art/Pokemon-Tileset-Fire-Red-Leaf-Green-Outdoors-C-446156842>

**Sprites**

Player sprites were custom-drawn by me, using the Tower Tycoon Palmer overworld sprite from Nintendo’s Pokemon Diamond/Pearl as a reference: <https://www.spriters-resource.com/ds_dsi/pokemondiamondpearl/sheet/6934/>. The design is based on a certainly blonde haired, sheika slate wielding, amnesic Hyrulian boy. That’s right. (Please don’t sue me Nintendo, I have too much to live for.)



Companion sprites were custom drawn by me, using the female Ace Trainer overworld sprite from Pokemon Diamond/Pearl as a reference: <https://www.spriters-resource.com/ds_dsi/pokemondiamondpearl/sheet/6931/>

**NPC images**

All NPC images were drawn by Ernesto Irawan: <https://www.artstation.com/ernestoirawan> and edited appropriately by me.

Fox: <http://pixeljoint.com/p/124605.htm>

**Background images**

Dojo background: Augustinas Raginskis, <https://www.artstation.com/artwork/3o19ZA>

Snow background: Reid Southen, <https://www.deviantart.com/rahll/art/Convergence-215267624>

Armoury background: Unknown

House background: Unknown

Lake background: <http://material.animehack.jp/bg_nature.html>, free to use

WASD keys: Thesaurusrex, <https://www.redbubble.com/people/thesaurusrex/works/21656630-wasd-keys-black?p=t-shirt>

**Dialog boxes**

All dialog boxes were made by me in Photoshop.

**Software used**

Tiled tilemap editor for creating and editing the tilemap: <https://www.mapeditor.org/>.

Pycharm 2018.4.2 (Professional Edition) for writing/debugging code and .xml files: <https://www.jetbrains.com/pycharm/>

Adobe Photoshop Elements 2018 for all image manipulation: <https://www.adobe.com/uk/products/photoshop-elements/>

**Code**

Pygame: <https://www.pygame.org/wiki/about>

TMX: <https://github.com/renfredxh/tmx> (for importing the tilemap)

PyQT: <https://wiki.python.org/moin/PyQt> (for the demographics GUI)

Pyllet Town: <https://github.com/renfredxh/pylletTown> (reference/jumping-off-point for the Game file code)

Text wrap: <https://www.pygame.org/wiki/TextWrap>

Input: <https://stackoverflow.com/questions/14111381/how-to-get-text-input-from-user-in-pygame> (jumping-off-point for the username input code)

I started this project on 23/01/2019 without any previous game dev knowledge and worked on it almost every day for multiple hours until 17/02/2019. Although it was definitely not always enjoyable (I despised the damn thing by the end of it) and it took the place of many, many other things I could and should have been doing, I have to say I’m quite proud of the result. Hopefully, you’ll agree 😊