DECO2800
25/10/2019

Individual Portfolio

DECO: COMBAT EVOLVED
STUDIO 1

Adestia Queenslandari
44806644

INDIVIDUAL

Overview

During the development of Deco: Combat Evolved (DCE), I mainly operated as a designer, working on general game design. My notable contributions include:

- storyline/game direction development (see wiki contributions here),
- user interface layout design (see here and here), and
- various asset designs (see here).

Nearing the end of the semester, I did some minor programming work, updating the old assets that were still in game.

Software and Tools

Design

Adobe Illustrator was collectively chosen by the designers due to most having used it previously, including myself. I felt confident in my ability to create assets with it and I am happy with the designs I created.

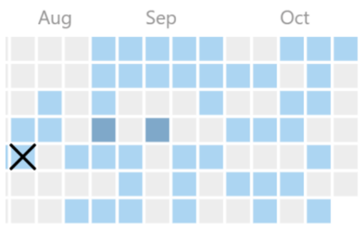
Programming

My experience with Java made it easier for me to pick up on how to implement things (once I had explored others' code). Gradle was used extensively to ensure changes made were passing all tests, even when just adding assets. Additionally, I play tested the game, visually checking that nothing seemed out of the ordinary. This process helped me to ensure code without JUnit tests was also working as intended. Because of this, I had not broken the build over the semester. However, on the final day of Sprint 5, I pushed to (and left) a broken build trying to update the end story backgrounds.

As all tests passed locally several times, play testing worked and I wanted to get my changes into the game before the Sprint was over, I thought that I was safe to push. However, I should have double checked the JUnit test (knowing it was failing randomly) and see what could have caused it to fail previously, instead of thinking I would get lucky. This is because there is no benefit to adding changes to a game when the overall game is broken and tests fail randomly.

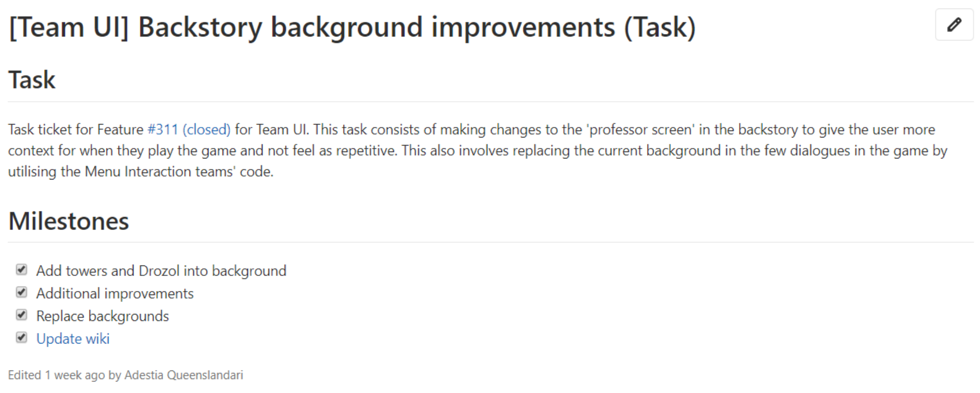
GitLab

I mainly talked with other teams during the studio, so I used GitLab more as a means of reaffirming things and checking out other teams' work.



Overall contributions on GitLab (tutorial exercise crossed out)

I also endeavoured to use labels, add additional comments, link wiki pages in my tickets for easy access and close issues where appropriate to let others know of my progress.



[Team UI] Backstory background improvements (Task)

Task

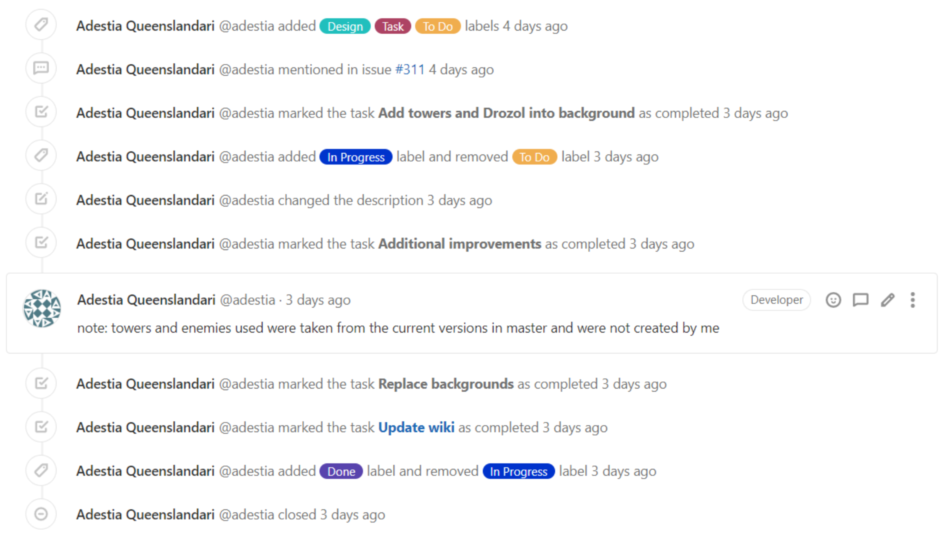
Task ticket for Feature #311 (closed) for Team UI. This task consists of making changes to the 'professor screen' in the backstory to give the user more context for when they play the game and not feel as repetitive. This also involves replacing the current background in the few dialogues in the game by utilising the Menu Interaction teams' code.

Milestones

- [x] Add towers and Drozol into background
- [x] Additional improvements
- [x] Replace backgrounds
- [x] Update wiki

Edited 1 week ago by Adestia Queenslandari

Main ticket format used



Adestia Queenslandari @adestia added Design Task To Do labels 4 days ago

Adestia Queenslandari @adestia mentioned in issue #311 4 days ago

Adestia Queenslandari @adestia marked the task **Add towers and Drozol into background** as completed 3 days ago

Adestia Queenslandari @adestia added In Progress label and removed To Do label 3 days ago

Adestia Queenslandari @adestia changed the description 3 days ago

Adestia Queenslandari @adestia marked the task **Additional improvements** as completed 3 days ago

Adestia Queenslandari @adestia · 3 days ago Developer
note: towers and enemies used were taken from the current versions in master and were not created by me

Adestia Queenslandari @adestia marked the task **Replace backgrounds** as completed 3 days ago

Adestia Queenslandari @adestia marked the task **Update wiki** as completed 3 days ago

Adestia Queenslandari @adestia added Done label and removed In Progress label 3 days ago

Adestia Queenslandari @adestia closed 3 days ago

Example use of GitLab features

Storyline Development

Sprint 1

I worked on the storyline heavily in Sprint 1. Initially, the studio was prompted to write game proposals. Although my original idea was not chosen, reading others and writing my own introduced me to what a fleshed out game entailed and some possible concerns/ideas other studio members had.

I focused on writing the backstory that the player would encounter when starting a new game and a short summary on how features integrated. I wrote a draft on a Google Doc and gained informal feedback from a few teams, mostly my own, to check that I was on the right track. Then, I sent the final storyline via Google Docs to the studio's slack for feedback. I didn't receive any comments and so I concluded that no one had conflicting ideas and went on to do user testing.

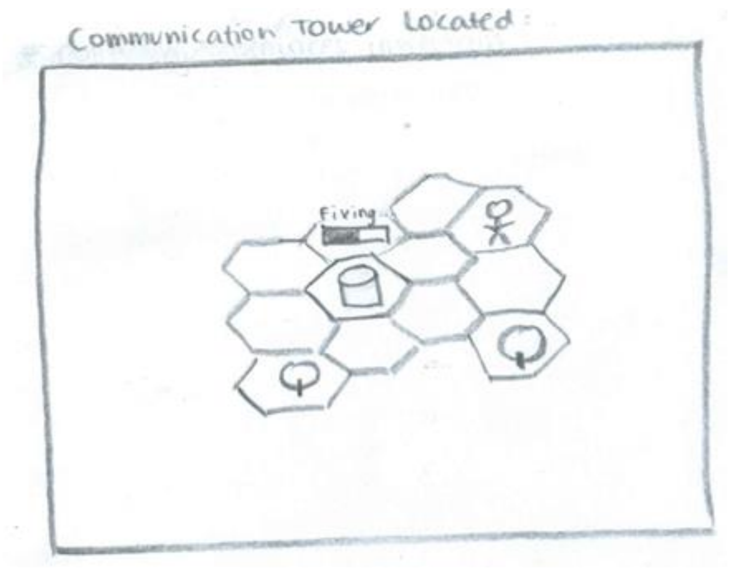
What I didn't realise was that almost no one read it, and at the start of Sprint 2, others had made different decisions about the game or didn't really care when asked. This was a mistake on my part. I should have been more proactive in the studio and enforced the game structure as soon as possible, instead of expecting people to read the wiki/slack and provide feedback voluntarily.

Additionally, I should have made a survey for studio feedback to reassess certain concepts I had assumed would align with everyone's features. This would also provide a layer of anonymity for members, potentially incentivizing some criticism, and being able to directly track the estimated number of people who have read the storyline (by viewing the number of responses). Then, I could also take measures if the number was too low.

Sprint 2

Due to the confusion in the game direction, I had the task of trying to make what everyone had already started implementing work with, instead of against, each other. To do this, I had to make some executive decisions on how features would fit into the game if the implementing team was unsure. I spoke with most of the studio and instead asked more definitive questions about their features, rather than just feedback on the proposed storyline.

From there I rehashed the game structure first and this time, sent it to all team's tickets, on slack and reminded everyone again during studio stand-up to cover all bases and ensure that everyone had a look at it. I also utilised sketches this time, to make sure that everyone completely understood what was meant by certain mechanics.



Visualisation of defense and communication tower mechanics in play

I believe this was a success and that I should have taken these steps in the first sprint to reduce the amount of time working on the storyline. Providing a game structure first and getting that approved, then moving onto the storyline would have also been the better way to go about this task. I believe the sketches also helped a lot with comprehension, as initially people were having a hard time distinguishing between communication and defense towers.

Sprint 3

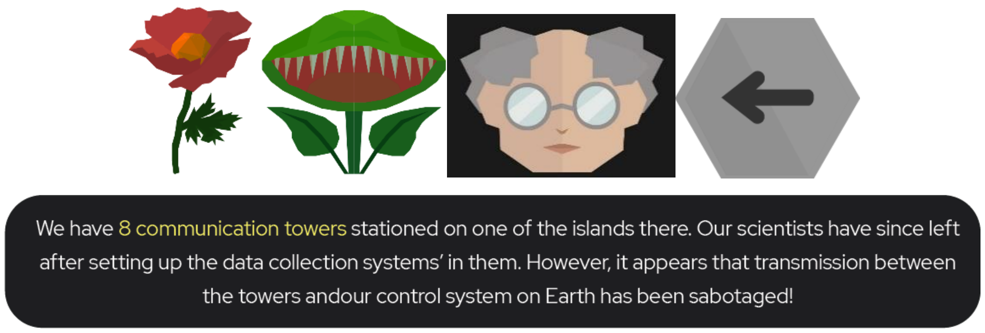
As a shared task with another member, I improved the storyline to better fit the new game structure and we created assets for it. This is discussed further in the Group Collaboration reflection.

Sprint 4

In Sprint 4, I wrote the end story. This was a shared task, but not as collaborative as in Sprint 3. As the end story was created after the game was well into development, it was easier to write one. This is discussed further in the Group Collaboration reflection.

User Interface/Asset Design

I helped develop the back/end story, lose and setting screens. For general asset designs, I made flowers for each biome and miscellaneous buttons and backgrounds for screens.


We have 8 communication towers stationed on one of the islands there. Our scientists have since left after setting up the data collection systems' in them. However, it appears that transmission between the towers andour control system on Earth has been sabotaged!

Examples of the variety of assets created

When designing, I generally followed the same approach:

1. Research patterns/designs and make sketches
2. Create high fidelity prototypes
3. Conduct user testing via Google Forms (typically static images of screens/assets)
4. Iterate and provide assets for implementation

By following this method, I was able to streamline my work and create personal deadlines for each sprint. In that way, I believe it was a good approach to designing. One of the fallbacks of this method is that I typically spent the whole sprint completing one design, providing my designs for implementation quite late in the sprint, taking a toll on the implementation team.

As an improvement, I'd reduce the time spent researching and testing. Although it helped me in making initial decisions, I believe that for relatively simple UIs such as a settings screen, it was not necessary to do extensive research and isolation testing due to these screens having high traffic and encountered in most play tests. By doing play tests instead, feedback on a variety of designs could have been collected more efficiently.

Alternatively, our team could have coordinated when we would make the UIs and then test them altogether instead of finding new participants each time.

GROUP

Description

My teams' focus was User Interface design. Once most of the UI was finished, we worked on improving placeholder/general assets that didn't fit the UI guidelines set.

Reflection

Task Allocation

At the start of a sprint, the team would brainstorm tasks to work on. One person would then make a base feature ticket with a summary of what we discussed. If there were any issues, we were free to edit the ticket.

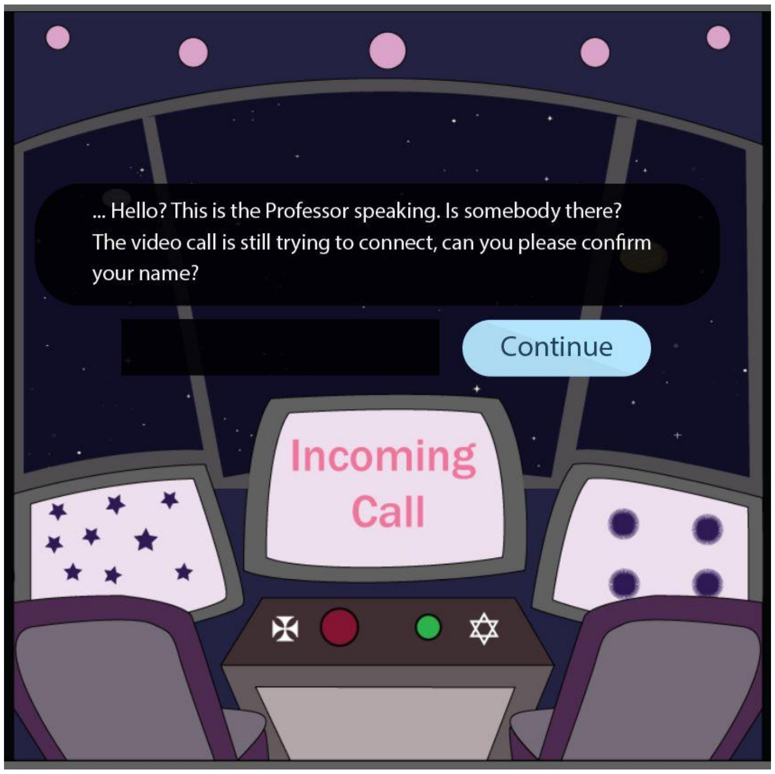
For sprints 1-3 this worked well, and we delegated tasks based on preferences. When there was a complex enough task, at most 2 people worked on it. However, for sprints 4 and 5, tasks became smaller and sparser. Hence, if someone had an idea for a task, they would be the one to take it on.

This led to task distribution being fairly even between members except in Sprint 5. This was when our studio session was cancelled due to the public holiday. This member didn't attend the next studio and so they were unable to scope out any tasks. Then, the small number of tasks that our team had found were already allocated. This led to them having almost no tasks to do.

Collaboration

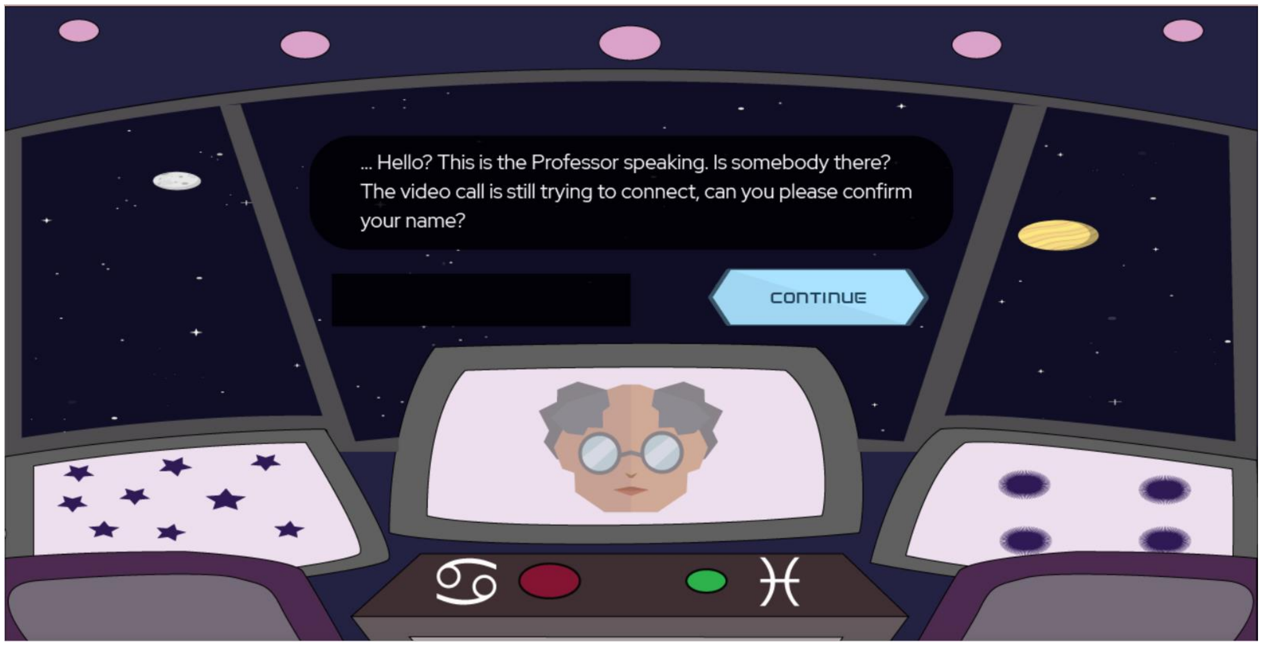
The team collaborated in a sense that we would ask for feedback or help from one another. Tasks were generally not shared due to their low complexity. If there were any, members discussed between themselves how they would allocate sub-tasks during the studio and then assign them in the task ticket. This allowed both parties to know who would be working on what and avoid duplicated work.

One issue I found in my first shared task was another member's work being done at a different rate/quality than my (personal) standard. They created a background asset and did not want to make many changes when I mentioned that the art style wasn't consistent with the guidelines or that it wasn't suitable for a typical monitor size. This led to me creating incorrectly sized dialogue assets to accommodate for that. We then combined them to get an idea as to what user testers would see, which is what I thought would be used.


... Hello? This is the Professor speaking. Is somebody there? The video call is still trying to connect, can you please confirm your name?

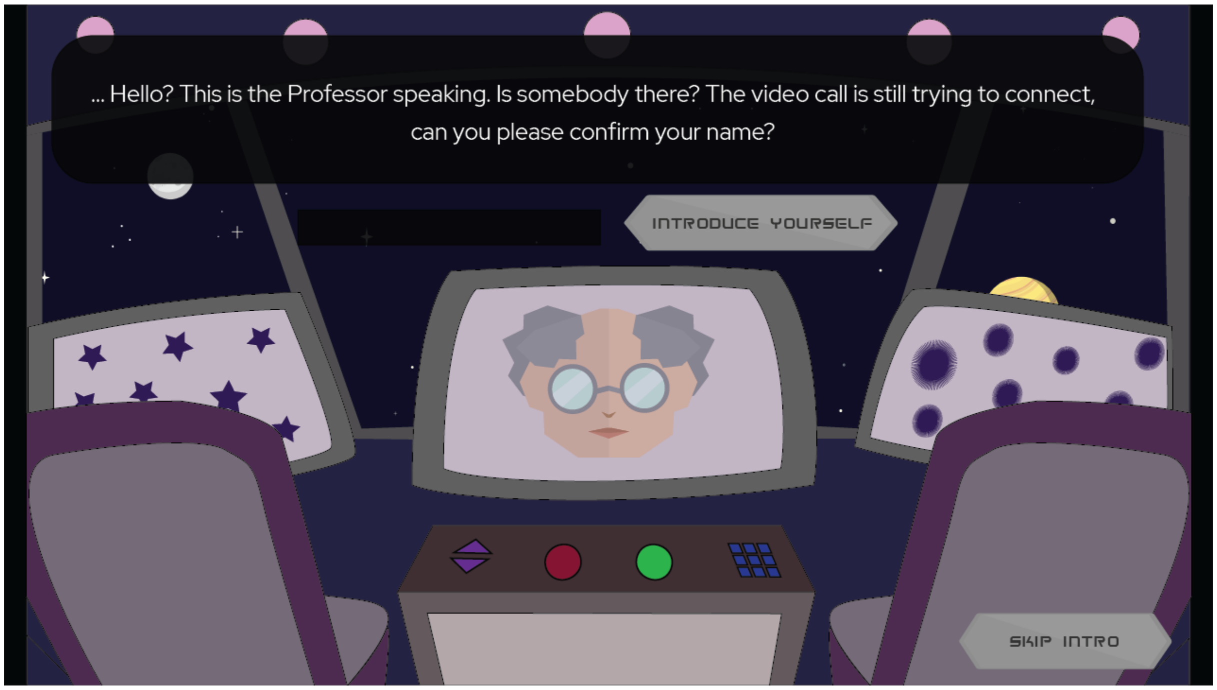
Example of our combined assets before user testing

However, when the other team member created their user testing prototype, they stretched the background anyways, and so the prototype did not look as cohesive as I believe it could have.


... Hello? This is the Professor speaking. Is somebody there? The video call is still trying to connect, can you please confirm your name?

Example asset used in testing prototype: stretched to fit the screen and dialogue smaller than intended

At the end of the sprint, I tried to overtake the task, as the other member hadn't updated their work yet and I was personally unsatisfied by the state of our assets.


... Hello? This is the Professor speaking. Is somebody there? The video call is still trying to connect, can you please confirm your name?

After asset improvements: resizing components, removing symbols and updating buttons

This is something we should have established at the start: expectations for time completion and quality. This would mean we weren't waiting for the other person to complete tasks before it was even possible for them to do (due to other assessment deadlines) and result in more cohesive assets in the first place.

Then in my second shared task creating the end story, I felt that if I spent too long writing a detailed story, my team member could not start their part. I tried to keep it succinct so that it would give the other member a decent amount of time to create all the assets and carry out user testing.

However, I now feel that I should have focused on writing a more satisfying resolution to the game and let my team member know to focus on other tasks in the meantime. In hindsight, this would have resulted in a better game experience overall. Again, setting up expectations for tasks would have helped avoid this.

Communication

The team conducted majority of our discussions during the studio, following up on slack for any issues. After allocating tasks, discussions were geared towards getting preliminary feedback on our designs before conducting formal user testing.

Though our communication was good overall, there were faults, especially regarding one of the team members. This member had been contributing sporadically, usually only on the last day of the sprint. A different member and I pointed this out to each other, however we did not directly confront them. I instead focused on my own tasks, which was easy due to the low coupling between our tasks. Looking back on this, I would try to communicate more with the team member, instead of not mentioning anything. This is an area of project collaboration I find the most difficult. I think that we would have completed more as a team if we had helped motivate the other person to contribute more consistently.