Change Report

Assessment 2

Team 10: Hard G For GIFs

Dragos Stoican

Rhys Milling

Samuel Plane

Quentin Rothman

Bowen Lyu

Summary: Changes to management

With the dawn of a new assessment, new documentation comes with it. This time, we are armed with previous knowledge of how we can improve our documents thanks to the previous assessment, seeing where we needed to improve. Overall, we saw that our documentation style was relatively fine, so we chose to stick to our guns in that department, bringing their documents and adapting them to how we work more efficiently. The only major difference we did is to lower our font size, so we could fit more information into the document.

There were no changes in how we handled documentation. We'd separate the team into two groups and have each handle a set of documents, drawing up an initial draft in notes form. Then, as a full team we go through them all and update them as we go. The only difference here is the documents we needed to do. We needed to make new documents but also adapt the old ones. We simply assigned one person to do that, rewrite what needed to be done, then review it as a team and make some final adjustments.

In regards to the code, it was rather different. Because while the documentation was complete, the code and the game wasn't as much. Notably, there were some core requirements that weren't implemented into the game. On top of that, we had the new requirements to implement. Immediately, we assigned two teams. One team would work on the new requirements, while the other would work on implementing the old ones while also refining the game's look and feel to match our style.

However, having multiple people working on the same code would draw mismatches in coding style, perhaps some inconsistencies, which was something that was noted last time as well. So, in addition to that, we decided to have one person go through the code and change the style to theirs, with the team being present as well to help out. Once that's done, team members would go through the code they made and make sure to comment the code appropriately, as well as edit the docstring to include all the new changes

Old Documentation revision/change report

I. Requirements

(*https://hardgforgifs.github.io/assessment-2/assessment2/Req1.pdf*)

The first change we made was the introduction of the document, where we elicit the requirements. The main issue with the previous introduction is that it spent too much time on how they came up with each specific section, what was what, etc. Meanwhile, they failed to properly explain what each table was meant to represent as a whole, and also didn't underline the meaning of priority, which we added as well.

Before they delivered their User Requirements table, they went through each of those requirements and described how and why they came up with them, as well as listing some alternatives for a few. We were confused as to why and simply decided to remove that part, not seeing the use of this, especially when it was only done for User Requirements and not for Functional or Non-Functional requirements. Whatever information that was useful there would be added to the tables.
Speaking of the tables, we also had to remake them in Google Docs, as they seemed to have used a third-party software to make them. The issue with their table is that we couldn't modify them. On top of that, they were hard to read due to being too small. This was likely due to how crunched for space they were, but getting rid of the earlier descriptions allowed us to gain more space to use.

Once the tables were modified, we now had to take a look at the content of these tables. As a team, we went through the requirements and decided that some had to go simply due to not making sense, or not being needed. Other requirements needed to be renamed, which we promptly did to better reflect their content. We made such changes to the User Requirements and Functional Requirements table.
After removing unnecessary ones, we brought over some old ones of ours that we saw could be useful, as well as creating new ones in order to match the new requirements we were given for this assessment.

II. Abstract and concrete architecture:

(https://hardgforgifs.github.io/assessment-2/assessment2/Arch1.pdf)

The approach when coding was not to modify most of what the other team has implemented, instead continuing their work in a similar manner by adding new elements to the architecture without having to apply major changes. We will cover every change made in the architecture document, then we will justify in detail the decisions we took.

The first section of the architecture document, consisting of the point 3a from assessment 1, was updated with the new UML diagrams that also contain the changes we made. In terms of the architecture, we have 7 new classes that are used to implement the power-up packs.

Abstract architecture:
In terms of the abstract architecture, our changes have been reflected in the Simplified Component Diagram section, as well as in appendix 1.

In the first picture, you can see the addition of the PowerUp class which inherits from GameObject and implements CollisionObject, much like the other classes in the same picture, which PowerUp resembles. The following block of text is modified to reflect the new change.

The following two pictures and their respective text aren't modified as we didn't have to change anything in the architecture of these elements.

We added one more picture to explain the inheritance used with the effects a powerup can have, as this functioned as a backbone of how we want effect to work in our game. We added the necessary text to briefly explain our decision, but a more detailed justification can be found here in the change report.

Their heavily object oriented approach fit very well with our plans of extending the project, so only these additions were made, no modifications or deletions.

Concrete architecture:
In terms of the concrete architecture, many changes were made to add new attributes and methods, as well as delete some old ones. However, overall the architecture is similar and works in the same way when it comes to inheritance and relationships, as we didn't see a reason to change it.

The first picture and it's brief explanation stay the same as there's no changes in the relationship between the scene classes and the main class.

In the second picture we added all the new attributes and methods related to the Scene objects. We also changed the ResultScreen class to a SceneResultScreen which implements the Scene interface. This is not a change we implemented, this was their implementation that was misrepresented in the architecture. Nothing else changed here.

In the section of inheritance diagram[3] we added the PowerUp and Effect classes, as well as the Effect subclasses to show the relationship between those and the BoatRace.

The last section doesn't include anything new other than the updated attributes and methods.

In the second section, point 3b from assessment 1 we added the justification relating to the new requirements in assessment 2. We explained our design choices briefly, and we are expanding on those decisions more here in the change report. We also decided to remove/condense some of the other team's existing content to make room for new content. We removed the justification for UR_FINALS_PLACING as this requirement was removed from the requirement table and so it was no longer relevant here.

The appendix of the document was also updated to match the new architecture.

Justification of new elements in the concrete architecture:

Power-up packs:
Requirements: UR_PICKUP_BOOST, FR_RANDOM_BOOST, FR_BOOST_DURATION
This is the biggest we made when it comes to the architecture and it consists of adding 7 new classes that implement the UR_PICKUP_BOOST requirement. We followed the inheritance used by the other team so we don't affect the overall structure of the code.

Therefore, the first new class needed is the PowerUp class. This class extends the GameObject class, because it requires textures and sprites to be displayed on the screen. Unlike some obstacles which have a similar implementation, this object doesn't extend the abstract moveableObject class because it doesn't need to move at any point. The class also implements the collisionObject interface, because it obviously needs to be collidable for the boat to be able to interact with it. Every PowerUp has a Effect it applies when the boat collides with it.

This brings us to the second new class we added to the code, the abstract Effect class. This class is crucial for applying an effect to a boat that is colliding with a powerup. This abstract class has a duration attribute, which is the remaining duration before the effect expires and stops affecting a boat, a Texture and a Sprite attributes that represent the asset that will be displayed on the screen when the player picks up a boost, and a is_active boolean attribute that is marked as false when the effect expires. The applyEffect method deals with applying the effect and also reducing it's duration on every frame. The getSprite method returns a sprite with an appropriate position to be displayed when the player boat is affected by this effect.

The 5 other new classes, SpeedEffect, Repair Effect, ManeuverabilityEffect, StaminaEffect and Invulnerability Effect extend the Effect and override the constructors to match the appropriate Texture and Sprite. On top of that, each of those objects also override the applyEffect method based on the type of effect.

This implementation makes it very easy to apply effects to boats, even multiple effects at once, and it also makes it very easy to display those on the screen. Therefore the only other changes that are reflected in the architecture for this feature are the updateEffects.

Saving the game:
Requirements: UR_SAVE_GAME, UR_LOAD_GAME, FR_SAVE, FR_LOAD
This feature is a complex one that is covered in detail in our implementation document. However, the changes reflected in the architecture are quite minimal. We had to add two new methods, saveGame and loadGame in the main class of the game, PixelBoat, along with the attribute "pref" that uses the Preferences library to save a game state.
New Batch attribute in the PixelBoat class:
However small, this change is very significant because we now have a new Batch "static_batch" that displays UI elements that are projected on the camera, instead being projected directly on the screen. This means we can use this batch to display static elements on the screen without them being affected by the player movement in the race.

New start game screen:

We improved the SceneStartGame class, by adding a new exit button screen and a load game button. These changes are reflected in the concrete architecture where we added the newly required textures and sprites.

New boat selection screen:
Requirement: UR_BOAT_SPECS
We improved the SceneBoatSelection class to make it look better, but also to be more intuitive for the player. It is now easier to tell which kind of boat to choose and the differences between them. These changes are represented in the concrete architecture by the newly added textures and sprites.

Pause game feature:
In the SceneMainGame class we added new textures and sprites and updated the update() method to implement a pause functionality. From the pause screen, the user can save and quit back to the main menu, so we needed new textures and sprites to implement a button with this functionality. This menu is displayed on the new static batch we made.

Changes to boat stats:
Now the boat object has a setSpec method which sets the new spec_id attribute of the boat, and the appearance and specifications of the boat based on this spec_id. Changes to max_speed, acceleration, maneuverability and durability_per_hit are made in the setStats method.

Changes to stamina:
We changed the regeneration of the boat to be delayed by a fixed amount of time after the boat stops accelerating. This changes are represented by the new attributes stamina_delay, time_to_recover and recovering in the Boat class

Added difficulty feature:
The boat class now has a difficulty attribute that affects the time to recover before stamina regenerates. This attribute is used in the setStats method to change the time_to_recover attribute. For the player, this attribute is set based on the choice made by the user in the boat selection screen, this choice is reflected in the difficulty_level attribute of the SceneSelectBoat class.

III. Methods and plans:

(https://hardgforgifs.github.io/assessment-2/assessment2/Plan1.pdf)

Well the entire document for this more or less had to be reworked due to us being 2 different teams that worked in fairly different ways. Our approach is the same as both teams are using the Scrum approach, and we do have some programs in common, but that’s about as much as there can be in common for a planning document of a different team for a different assessment.

We redid the Method & Tool selection. We added on the different programs that we used, and removed the ones we didn't use because why would we keep them on. Our introduction also goes into more detail about the main programs we used to work and communicate with each other; while also listing additional programs we used, even briefly, but ended up being important to our project overall.

The Team organisation also had to be entirely redone. While both teams used the Scrum approach, our team had its own variation of it, and it needed its own justification. Obviously, we weren't going to change our style much as we did relatively well in term 1.

The plan obviously needed to be entirely redone. Not only was it the plan for Assessment 1, it was theirs, which one would only assume wouldn't be the same plan we had. As such, the Gantt chart was replaced with our one, and we went in detail about our plan during the weeks leading up to the final submission date.

IV. Risk assessment and mitigation: approach, presentation, risks, mitigations

(https://hardgforgifs.github.io/assessment-2/assessment2/Risk1.pdf)

Their Risk document is probably the one we made the least changes to, as it was overall rather solid. The main changes we made was rewriting parts of the introduction, as we felt it was rather lacking in explanations of the scales and layout of the table. Other than that, the document stayed relatively similar, with maybe some risks being added or removed as the team saw fit. We did have to remake the table itself but didn't need to change anything about it.
This is one of those documents that no matter who makes the expansion of the game, there shouldn't be any changes unless something drastic changes in the management of the project. Since we aren't supposed to consider stuff like bugs or the product not functioning as risks, and the conditions of the assessment are different, nothing had to be changed.