Method Selection & Planning

Mario

GROUP 28

Joseph Frankland

Anna Singleton

Saj Hoque

Leif Kemp

Shi (Lucy) Li

Hugo Kwok

a. Development and Collaboration Methods/Tools

Collaborative Tools

The most common version control tools we found were Git, CVS (Concurrent Versions System), and SVN (Apache Subversion).

	#1 Git	#2 CVS	#3 SVN
Pros	- The team members were already familiar - Has an easily accessible, well-designed website for hosting - GitHub gives clear overviews for contributions, edit history, collaboration, and more - Large community with many useful resources for help - Has both CLI and GUI - Easy to branch and fork	- Been around a long time and is trusted	- Utilizes file locking to prevent multiple editing the same file at once - Highly configurable
Cons	- Merging conflicts can be confusing - Cannot move or rename files (it just gets deleted and added as a new file)	- Changes are time consuming - Doesn't support atomic commits - Doesn't support merge tracking - Doesn't handle distributed source control well	- Time consuming to commit - Difficult to learn - Doesn't store file modification times

We decided to use Git for our version control as members of the team were already familiar and we were provided with lectures as well to help the members of the team less familiar.
For collaborating on deliverables and constructing our plan we decided to use Google Drive, this was because it had the simplest tools for working together, such as being able to see the edit history so we could revert changes if necessary whilst also being able to work in either a synchronous or an asynchronous environment. Alternatives such as Dropbox were considered, however ease of access was our deciding factor when it came to transcribing deliverables. This option was quickly disregarded due to the convenience of Google Drive linking to our members' university accounts.

Integrated Development Environment

Of the available IDEs, we looked at the main three: Eclipse, IntelliJ IDEA and VSCode. They are all similar and support the following features (non-exhaustive list):

- View JavaDoc by hovering over the symbol
- Auto-complete (automatically fills in the remaining line)
- Auto-suggest (shows a list of useful options as you type)
- Auto code formatting (automatically format and style the code)
- Auto imports (handles importing everything required in the classes)
- Git integration with commit history graph and comparator
- Debug tools including breakpoints and code stepping

All programs provided all the tools we needed for writing, testing, and debugging the code. We decided to use VSCode. This was because the members of the team doing the majority of development had previous experience with the IDE, therefore, it seemed more preferable as it removed the requirement of learning how to use a new IDE.

Game Engine

One member of the team was already familiar with Java game development and available game libraries. A few of the options were LWJGL, Slick2D, and libGDX.

- We decided against using LWJGL as it would involve coding for OpenGL directly.
- Slick2D is a lightweight 2D game library that is built on LWJGL but is very easy to use and understand.
    - However, it hasn't been updated since 2015 and some of the dependencies has since stopped being included in the Java Development Kit, making it unusable for our purposes.
- We decided to use libGDX as it is also easy to understand.
    - It is being actively developed.
    - Based on LWJGL but removes the need to write OpenGL directly.
    - Although the javadoc is not great, libGDX has a helpful community and wiki that assists developers in understanding how to use the platform.
    - There are multiple addons and integrations for libGDX which allows developers to easily add features. For example,
        - It supports Box2D which is a physics engine,
        - GDXVideo which allows video playback within game,
        - And an AI platform which allows for computer control of objects.

Communication

- To communicate with each other outside of the in-person sessions, our team decided to use a Discord server.
- Discord is a service that most of us were already using, so this was a large factor in choosing this method.
    - This removed the need to learn a new workflow by switching to a different tool, which would make communication more difficult.
- Alternatives included Teams and Slack which are similar to Discord but are more for corporate settings.
    - Slack is very similar to Discord but it places some of its features behind a paywall.

Methods

- Our team used the Agile methodology as a basis for our working scheme and how we would lay out the project.
- We chose this method because of the simplicity and time restraints of the project.
- The developers verified and tested each other's code to check for obvious errors.
- We also collaborated in calls to work together on solutions.
- The agile method is ideal as we would involve the stakeholders throughout development

Other Tools

Whilst working on this project we came across other tools we found necessary:

- Plant UML
    - This tool was used to create our concrete architecture diagram, it has built in tools that makes creating class diagrams easier(ie. showing relationships between classes, using symbols to represent visibility of methods and variables, etc.).
- Trello
    - This tool was used to base out our plan throughout the project, its significance in usage defined by its collaborative abilities. We talk more about this in the organisation.

b. Organisation

- Tasks are delegated to different members of the team during the in-person sessions held every Friday. They are then tracked on a Trello board and given a date which they should be completed by.
- The team has been divided into 2 groups, one group working on the implementation and the other working on the deliverables for the project. Two people, Anna and Leif are the two working on the development aspects of the game whilst the remaining members are on the deliverables team.
- In terms of leaders there isn't a defined one. We decide amongst ourselves what we want to do and then assign ourselves the tasks on the trello board.

We decided to use a Trello board for management of tasks as it's easy to use and read. As well as this it allows you to comment on different tasks if you are stuck and need help with something specific. Setting due dates for tasks is helpful as close to when it is due you will receive an email notification which reminds you that the task needs to be done as soon as possible. Trello boards also allow you to set a priority rating of a task which can let others on the project know what needs to be done first and what things can be left until later.

Dividing into two groups seemed like the most logical idea, this decision being influenced by basing each member's preference and which department they were best suited for, allowing a more efficient approach to the project. This makes each team easier to manage as well as meaning not every member has to worry about both sides of the project. However we've ensured regular collaboration between the two sub-groups for clarity in our work as well as necessary information that is relevant to each group. This was approached through weekly meetings that involved each team updating the other with their progress including any questions or concerns that they may have. This allowed fluidity between the different sides of the project which in turn helped support a dynamic work environment. It also helps with time management as we can do the implementation and deliverables work simultaneously, cutting down on time of waiting to complete some of the documentation after implementation if that was necessary. With this approach lower priority tasks can be completed whilst waiting for specific implementation to be completed.

Not having a defined leader was a decision we made as a group. It means less assignments of work will cause conflict as people are mostly able to get what they want to do. This can be achieved through the functionality of designating tasks in Trello. Furthermore, it allows for the ability to work dynamically as once you complete a task or get stuck on another you can move to a different one until you're able to make some progress on the previous task.

The methodology we've decided to use for the management of our project is the agile workflow. When we meet up on a Friday each team will start a sprint, this is usually either one or two weeks depending on the size of the task. Tasks will be assigned via each respective Trello board. Once a sprint has finished we will review the work and then make changes if necessary and then start the next sprint for the next part of the project.

c. Plan

Task #	Task	Priority	Start Date	End Date	Assignee
1	Review Existing Systems	HIGH	Nov 26 2021	Dec 3 2021	Anna Singleton
2	Stakeholder Interview	HIGH	Nov 29 2021	Nov 29 2021	Everyone
3	Gather Requirements Together	HIGH	Nov 29 2021	Dec 2 2021	Everyone
4	Design	HIGH	Dec 2 2021	Dec 17 2021	Anna Singleton, Hugo Kwok
5	Setup libGDX	HIGH	Dec 2 2021	Dec 4 2021	Everyone
6	Splashscreen	LOW	Dec 4 2021	Dec 8 2021	Leif Kemp
7	Resource Manager	HIGH	Dec 4 2021	Dec 15 2021	Anna Singleton
8	Entity Class	HIGH	Jan 6 2022	Jan 9 2022	Anna Singleton
9	Ship Class	HIGH	Jan 9 2022	Jan 13 2022	Leif Kemp
10	Control system	HIGH	Jan 13 2022	Jan 15 2022	Leif Kemp
11	Cannonball System	HIGH	Jan 13 2022	Jan 15 2022	Joseph Frankland, Sazidul Hoque
12	College Class	HIGH	Jan 9 2022	Jan 11 2022	Anna Singleton
13	World Class	HIGH	Jan 10 2022	Jan 19 2022	Leif Kemp
14	Minimap System	LOW	Jan 19 2022	Jan 22 2022	Leif Kemp
15	Collisions	MED	Jan 19 2022	Jan 21 2022	Anna Singleton
16	NPC Ships	MED	Jan 19 2022	Jan 24 2022	Anna Singleton
17	College Placement System	MED	Jan 19 2022	Jan 20 2022	Leif Kemp
18	Win Condition	HIGH	Jan 20 2022	Jan 25 2022	Leif Kemp
19	Debug System	MED	Jan 6 2022	Jan 8 2022	Anna Singleton
20	Bad Weather	HIGH	18 Feb 2022	25 Feb 2022	Leif Kemp
21	Obstacles (mines)	HIGH	18 Feb 2022	25 Feb 2022	Leif Kemp
22	5 Special Powers	HIGH	25 Feb 2022	4 Mar 2022	Anna Singleton
23	Update Controls Screen	LOW	25 Feb 2022	4 Mar 2022	Hugo Kwok, Lucy Li
24	Difficulty Settings	MED	25 Feb 2022	4 Mar 2022	Leif Kemp
25	Spend Plunder	MED	8 Apr 2022	10 Apr 2022	Anna Singleton

26	Continuous Integration	MED	11 Mar 2022	25 Mar 2022	Leif Kemp
27	Combat with other Ships	HIGH	25 Mar 2022	15 Apr 2022	Anna Singleton
28	Save Files	HIGH	15 Apr 2022	25 Apr 2022	Anna Singleton
29	Edit documents	HIGH	18 Feb 2022	22 Apr 2022	Sazidul Hoque, Hugo Kwok, Joseph Frankland, Lucy Li
30	Change Report	HIGH	01 Apr 2022	29 Apr 2022	Sazidul Hoque, Hugo Kwok, Joseph Frankland, Lucy Li
31	Testing	HIGH	25 Apr 2022	29 Apr 2022	Everyone
32	Continuous Integration Report	HIGH	22 Apr 2022	30 Apr 2022	Sazidul Hoque, Hugo Kwok, Joseph Frankland, Lucy Li

The more detailed trello board of this plan can be found on the team's website: http://www.shardsoftware.tk/progress

How the plan changed over the course of development

- Our initial plan was not very detailed.
- The first plan was very rough and only consisted of abstract tasks.
    - This included things like 'implement front/back end' instead of specifics.
- After we had got a design together, we changed the plan to include specific tasks and their dependencies as seen above.
- During development we realised we had missed some of the dependencies so we modified the plan to show the new dependencies.

Comments

- There is a large gap between mid-December and January - this was to account for the Christmas break as we didn't expect people to be contributing during that time
- We added a splash screen as we found there was a delay while loading the assets
    - This provides the user something to look at while the content loads
- Because the world is randomly generated, we decided to include a minimap.
    - This wasn't part of the requirements but it helps during development as we can see the overall map.

All of these game features are set out at the start of the implementation stage. From this point the developers combine to propose the significance of each feature in relation to the game - allowing for priorities to be set with respect to the different tasks available. Collectively this will give the developers an idea of what tasks should be given more time and what should be done first. This also prevents the developers from wasting their efforts on a feature that may not be regarded as important to the whole development process.

We can see how our agile methodology is present throughout our plan with our various tasks being split into sprints every week or two - this has also given us the opportunity to regularly review our work.