Methods and Planning

Group 29
Shard Software

James Burnell
Hector Woods
Jensen Bradshaw
Ben Faulkner
Adam Leuty
Jiahao Shang

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.

Integrated Development Environment

Of the available IDEs, we looked at the main two: Eclipse and IntelliJ IDEA. They are both 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
- Lists class fields and methods on the side for navigation
- Task lists (displays flagged comments in a task list)
- Debug tools including breakpoints and code stepping

Both programs provided all the tools we needed for writing, testing, and debugging the code. We decided to use mainly Eclipse but one team member decided to use IntelliJ. We firmly decided against using VSCode as it is harder to set up, use, and lacks many features present in IDEs that were designed for Java development.

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 a combination of the Agile and Waterfall development methods. We met with the stakeholder once to form a requirements document but then used the document as reference. We then changed our plans and designs as we developed the code.
- 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, however,
- The waterfall method allows us to develop the code quickly without needing to wait for responses.

b. Organisation

- Tasks are delegated to various people in the group during in-person meetings and tracked on a spreadsheet.
- The team can be roughly divided into two groups; one working on the game and one working on the deliverables.
- Two or three people will work on the code while everyone works on the deliverables.
- We decided that the team leaders should be the developers, Hector and James, as they are the most familiar with the code and can guide the rest of the team based on that.
    - The team leaders will communicate with team about what needs to be done and when

There are multiple stakeholders involved in the development of this project. One of these is the customer. This stakeholder is where the majority of the game's requirements come from, being derived from a meeting via video call at the start of the project. Since he is interested in trying to sell the game, he is the most invested in its completion. After the initial meeting, communication with this stakeholder has been done mostly through email to ask for clarifications and input.

The project's other stakeholder is the University of York Communications Office. Communication with this stakeholder is more difficult, as it can only be done by proxy through the lecturers, so we have not had as much conversation with them. As such, this stakeholder has a less direct impact on the development of the product; however, their needs have also been considered.

c. Plan

Task #	Task	Priority	Start Date	End Date	Dependencies
1	Review Existing Systems	HIGH	Nov 26 2021	Dec 3 2021	
2	Stakeholder Interview	HIGH	Nov 29 2021	Nov 29 2021	
3	Gather Requirements Together	HIGH	Nov 29 2021	Dec 2 2021	2
4	Design	HIGH	Dec 2 2021	Dec 17 2021	2, 3
5	Setup libGDX	HIGH	Dec 2 2021	Dec 4 2021	1
6	Splashscreen	LOW	Dec 4 2021	Dec 8 2021	5
7	Resource Manager	HIGH	Dec 4 2021	Dec 15 2021	5
8	Entity Class	HIGH	Jan 6 2022	Jan 9 2022	7
9	Ship Class	HIGH	Jan 9 2022	Jan 13 2022	8
10	Control system	HIGH	Jan 13 2022	Jan 15 2022	9
11	Cannonball System	HIGH	Jan 13 2022	Jan 15 2022	9
12	College Class	HIGH	Jan 9 2022	Jan 11 2022	8
13	World Class	HIGH	Jan 10 2022	Jan 19 2022	7
14	Minimap System	LOW	Jan 19 2022	Jan 22 2022	7, 13
15	Collisions	MED	Jan 19 2022	Jan 21 2022	8, 13
16	NPC Ships	MED	Jan 19 2022	Jan 24 2022	9, 13
17	College Placement System	MED	Jan 19 2022	Jan 20 2022	12, 13
18	Win Condition	HIGH	Jan 20 2022	Jan 25 2022	11, 17
19	Debug System	MED	Jan 6 2022	Jan 8 2022	5

The Gantt chart 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 realized 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 splashscreen 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.