

Group 3 Project Report

In submitting this team project and report, we confirm that our conduct during this project adheres to the Student Code of Conduct. We confirm that we did NOT act in such a way that would constitute cheating, misrepresentation, or unfairness, including but not limited to, using unauthorized aids and assistance, personating other persons, and committing plagiarism.

Sharmarke Kedie _____ 

Dignesh Solanki _____ 

Rohan Limbachia _____ 

Nasir Giwa _____ 

Khalid Al Fattouhi Al Jundi _____ 

Contents

Group 3 Project Report	1
1. Introduction	2
1.1 Purpose of project	2
1.2 Issues to be discussed and their significance	2
1.3 Project methods	3
1.4 Limitations and assumptions.....	4
2. Project Management with Scrum	4
Use cases as backlog items	4
2.2 Creation of tasks	5
2.3 Planning and Carrying out the Sprint	6
2.4 Team management	7
3. (Object-Oriented) Software Development	7

Identification of classes.....	7
Software design with class diagram(s)	8
3.3 Techniques of implementation.....	9
3.4 Software testing and operation	10
4 Discussions and Recommendations	10
7.1 Discussions about the team projects	10
7.2 Recommendations for better practices	11
Conclusion	11
Appendix A List of Backlog Items and Tasks with Brief Descriptions	12
Appendix B Signatures for Contribution Table (with table attached in excel file)	14
Appendix C Signatures for Participation Table (with table attached in excel file)	14

1. Introduction

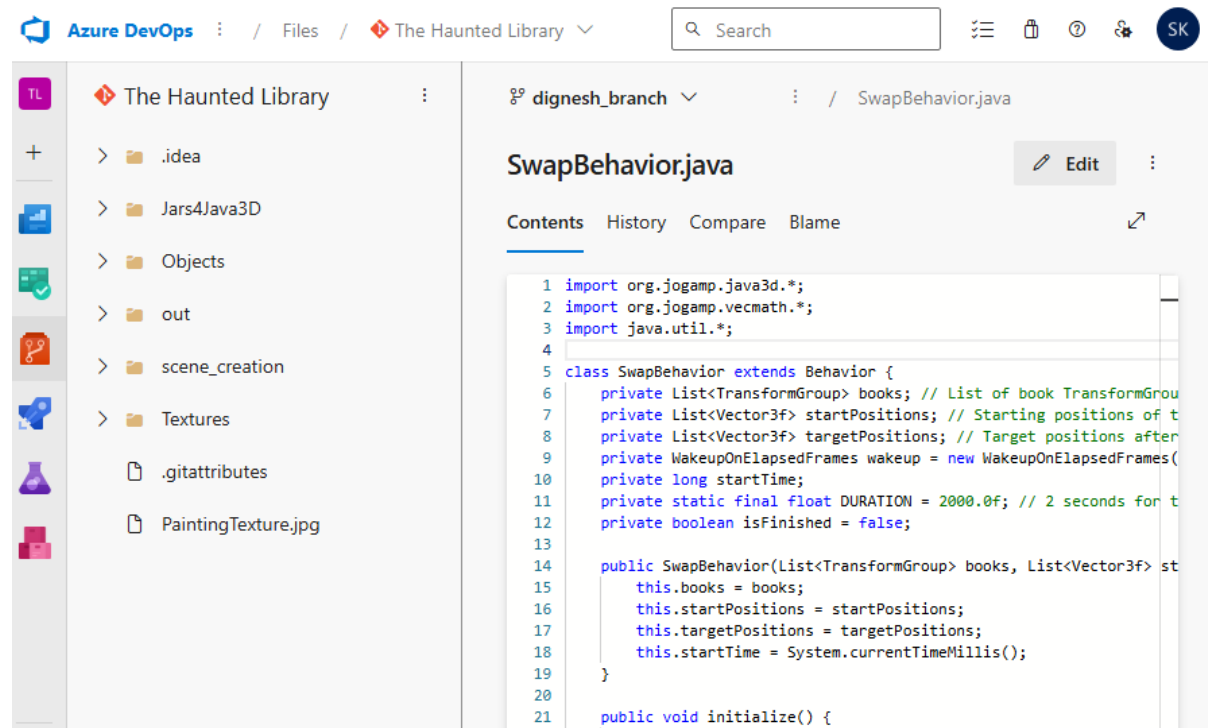
1.1 Purpose of project

We set out to create the 2nd floor of Leddy Library. The context of the game is this. You are trapped in Leddy Library and you must get out of Leddy Library before the time runs out. If you do not, then a ghost will appear, and the game will be over. To get out of Leddy Library, you need to complete 8 tasks related to organizing books. These tasks vary in level of difficulty. You have a maximum of two tries to complete one of the 8 tasks. We had two objectives. The first was to recreate Leddy Library and allow for a realistic experience. The second was to make Leddy Library haunted through sound design and spooky games.

1.2 Issues to be discussed and their significance

There were several challenges we faced. First, we needed to decide on what VR application we'd make. It was important that all the members of the team agree on a project before starting. Second was the project management with Scrum. Which includes making sprints, adding backlog items, updating tasks in the backlog items, setting up meetings and managing the whole process. This was challenging because most of our team had little to no experience using Azure (required Scrum tool) and practicing Scrum. The third thing we needed to do was include all the required features as backlog items. Then we split the responsibilities between the team members. Each

team member was responsible for a specific set of requirements and how to implement it into the game. We were then able to start designing the different classes that would act like information experts for each of the different responsibilities. This was challenging as we wanted to ensure we were following object-oriented design principles (like low coupling and high cohesion), but many of the team members did not have good understanding of OOP. Then came the hardest part which was implementation. Since we only learned java3D this semester it was hard to implement what we needed to do at the start. After adding a backlog item, we'd test it with the existing backlog items. This was made possible using git repositories. Each team member had a branch, and they'd push to remote whenever they were completed with the task. Then we'd merge with the main branch, and test to see if the feature worked when integrated (as seen below).



If not, then we'd need to modify the code and test again.

1.3 Project methods

The project adopted the iterative approach characteristic of the Unified Process (UP). Each sprint focused on completing a subset of requirements. This resulted in a functional demo at the end of each sprint, providing insights into capabilities and limitations based on our understanding, timeframe, and the chosen technology (Java3D). Each demo cycle involved activities from across the UP phases, though the emphasis shifted. The initial demo emphasized inception and requirements analysis, involving basic design and implementation for core features. This design was then implemented and tested. The primary goal of each sprint was to produce these

incremental demos. The final sprint placed greater emphasis on refining the design and comprehensive testing.

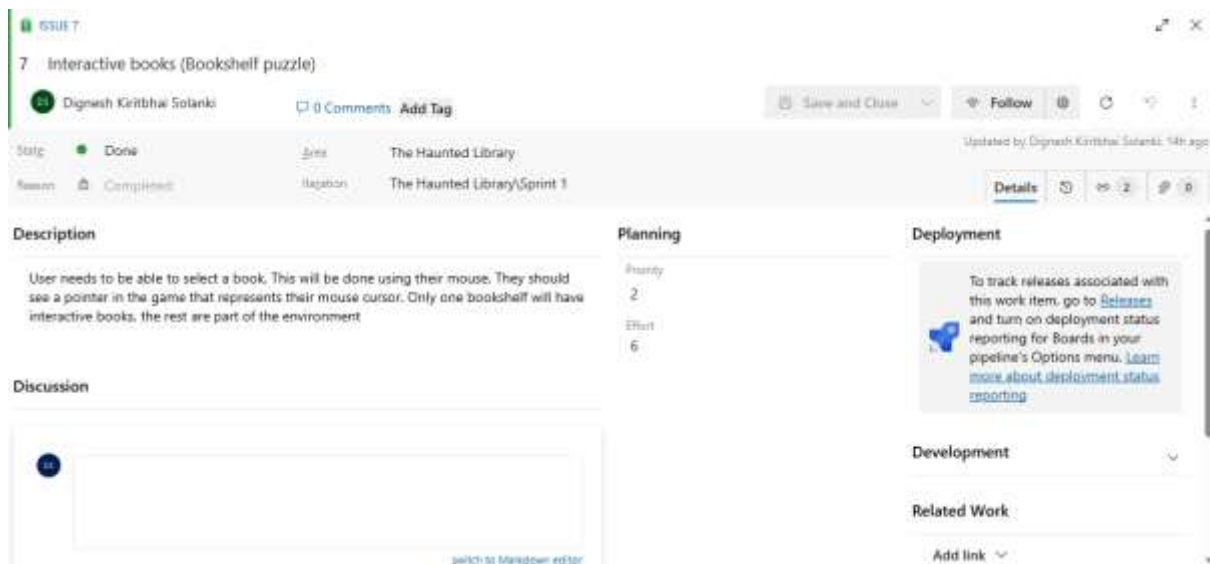
1.4 Limitations and assumptions

From the outset, the team established realistic expectations regarding project scope. Given the one-month timeframe, the focus was prioritized on core game features, with additional functionalities considered time-permitting. Priority was given to fundamental requirements, such as door interactions (requiring structural changes in the virtual environment) and keyboard-based movement, due to uncertainty in task duration estimates. Identifying and leveraging individual team member strengths was also a key consideration during planning.

2. Project Management with Scrum

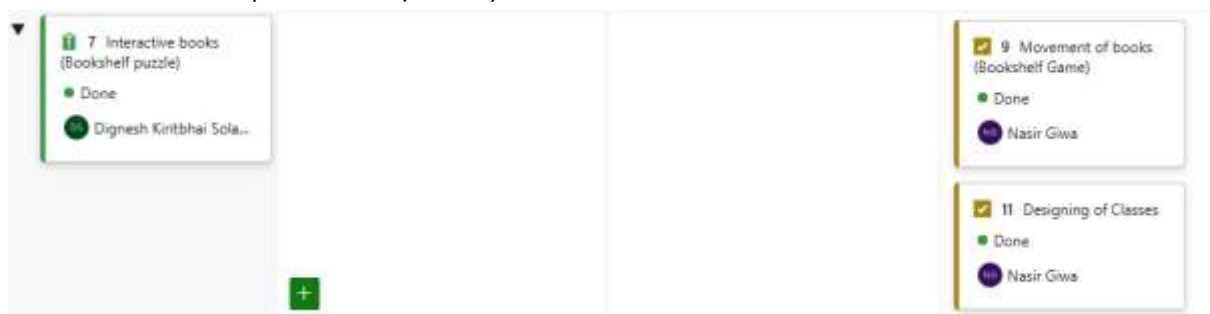
Use cases as backlog items

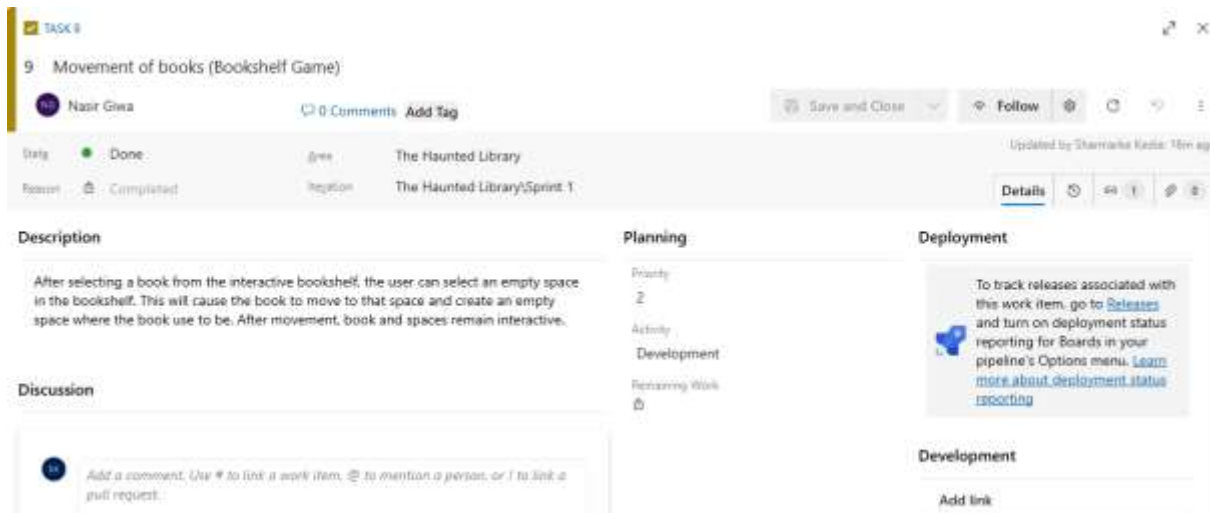
Each sprint consisted of a collection of backlog items that needed to be completed. These backlog items contained a set of tasks that are meant to fulfil the requirements for the project. The set of required features were outlined by the professor. Who represented the product owner. Through our weekly meetings, he provided feedback on what he expected. The development team was responsible for taking the requirements and implementing it into the game. For example, the professors required that the books be more realistic, so we added a variety of book covers which better match a typical library. We would create a brief use case description for each of the backlog items so that team members has a better understanding of what they needed to do. Here's an example of one of the backlog items.



2.2 Creation of tasks

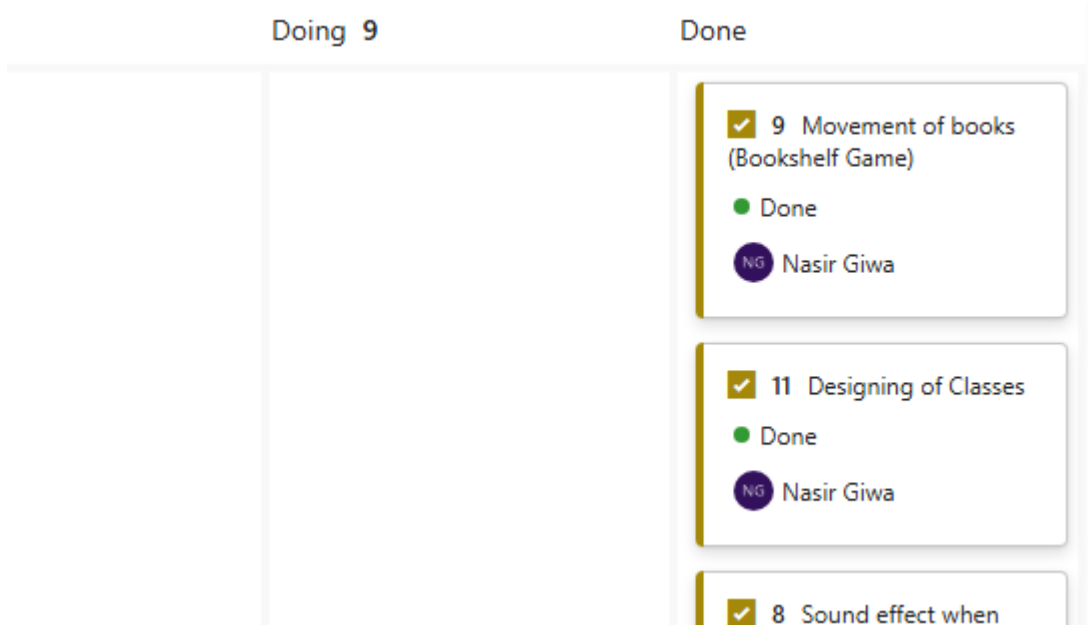
Each backlog item consisted of a set of tasks that were needed in order to complete it. A member of the team was responsible for each of the backlog items. However multiple team members have worked on the tasks of one backlog item. The additional members were usually helpers that either helped integrate the tasks together, or aid when needed. Sometimes backlog items were similar enough that team members need to work together in order to complete the tasks. For example, the *Creation of 3D Objects* and *Creating the library* backlog items. Since the *Creating the library* backlog item relied on the tasks done in *Creation of 3D Objects*, the team members responsible for these items often had to communicate. If the bookshelves created in blender didn't render properly in the java3D program then the person responsible for the *Creation of 3D Objects*, task would have to fix that. Here are the tasks that were completed for the *Interactive books (Bookshelf puzzle)* task:



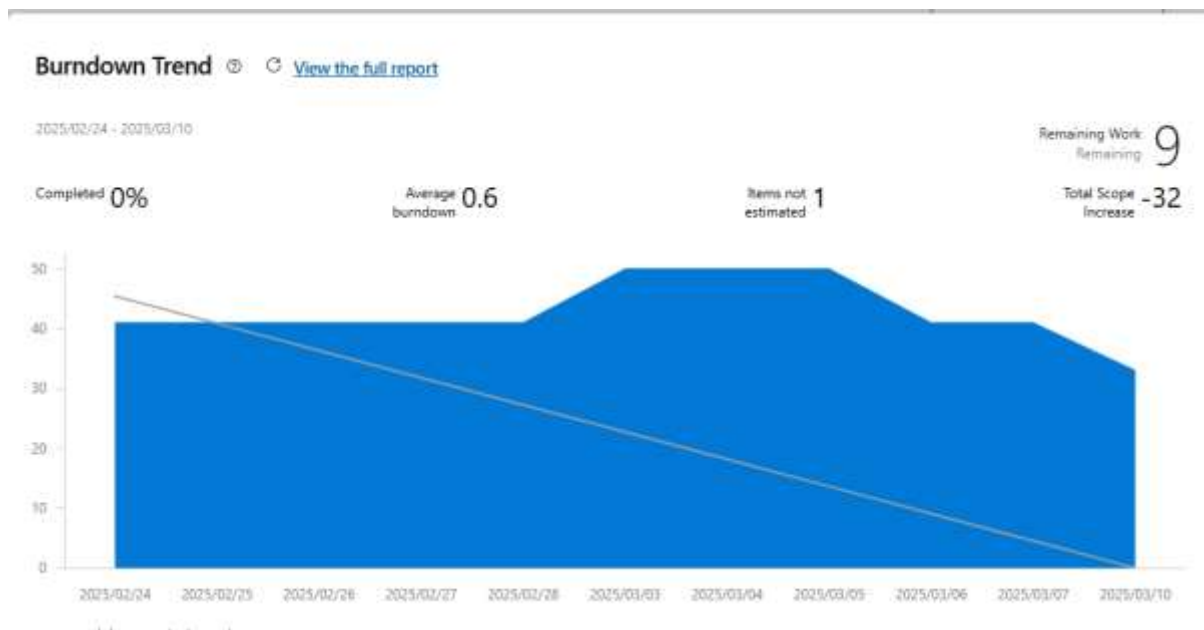


2.3 Planning and Carrying out the Sprint

We planned sprints in 2-week blocks. Since we only had a month. We only completed two sprints. The first sprint included all the necessary features like creating the shelves, the books, the movement of the user, defining the viewer, and making a simple game. Whenever a team member completed a task, they placed the amount of effort they did and moved the task to done.



Placing a task as done meant that the burndown chart was updated. The burndown chart helped us to keep track of how much work we had, and whether or not we were on track to complete it.



After each sprint, we had a sprint review with the professor (product owner) where we reviewed a demo we made. He provide us feedback to use in our second sprint.

2.4 Team management

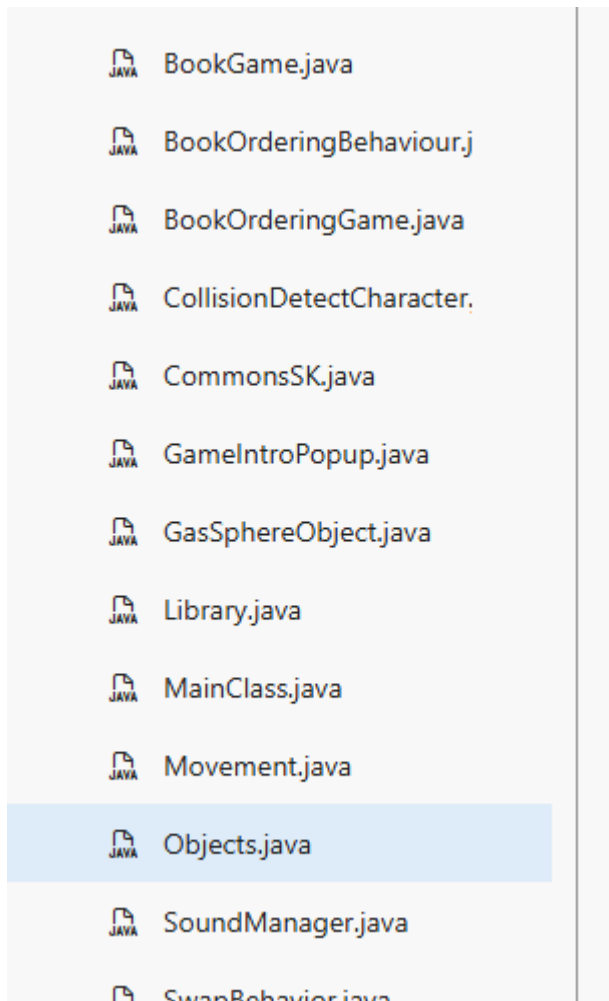
The Scrum Master facilitated team meetings, which were held twice weekly at mutually agreed-upon times. Meetings were conducted predominantly online, supplemented by occasional brief, in-person stand-up meetings. Regular meetings with the professor (Product Owner) provided valuable feedback and direction. The Scrum Master established the Git repository and workflow to ensure smooth integration of code changes. Task assignments were agreed upon during Sprint Planning to prevent duplication of effort. A WhatsApp group chat served as an informal communication channel for addressing immediate issues and coordinating meetings.

3. (Object-Oriented) Software Development

3.1 Identification of classes

The Information Expert GRASP pattern was the primary principle guiding class responsibility assignment. For game objects, polymorphism was implemented using a base Objects class (or similar) with subclasses for specific object types (e.g., HandleObject for door handles). Other classes were also designed following the Information Expert pattern, ensuring each class had a distinct responsibility. For example, a SoundManager class encapsulated all sound-related functionalities used by the main application class (MainClass). MainClass orchestrated the interaction

between different components to form the complete game.



3.2 Software design with class diagram(s)

We used Visual Studio 2015 to create the class diagrams for the different classes. There was mostly association between the classes. Generalization is used for the object class and it's inheriting class.

3.3 Techniques of implementation

The project required that we implement the classes in java3D. Which is Java but with more libraries. After creating a rough draft of the class diagrams, we started implementing all the methods. Once we started adding more and more methods, the classes got too bloated, so we needed to make artificial classes using the pure fabrication design pattern. An example would be the CollisionDetectCharacter class. Before that class was created, collision detection was defined in the MainClass. This led to low maintainability since changes in one part of the class would lead to changes in other areas of the MainClass. As we continued with the sprints, we increased the number of classes to accommodate the new features added. A simple example would be adding more classes that were inheriting from the Object class like DoorObject. OOP (writing the code) improved our understanding of how to design the system better so that there was increased modularity and scalability. Having separate classes where each team member worked on a class led to easier merging between branches, which led to a smoother workflow and thus increased productivity.

```

1 package scene_creation;
2
3 import java.io.FileNotFoundException;
4 import java.util.*;
5 import org.jogamp.java3d.*;
6 import org.jogamp.java3d.loaders.*;
7 import org.jogamp.java3d.loaders.objectfile.ObjectFile;
8 import org.jogamp.java3d.utils.geometry.Box;
9 import org.jogamp.java3d.utils.geometry.Primitive;
10 import org.jogamp.java3d.utils.geometry.Sphere;
11 import org.jogamp.java3d.utils.image.TextureLoader;
12 import org.jogamp.vecmath.*;
13 import java.util.Iterator;
14 import java.util.Random;
15
16 public abstract class Objects {
17     protected BranchGroup objBG;           // load external object to 'objBG'
18     protected TransformGroup objTG;         // use 'objTG' to position an object
19     protected TransformGroup objRG;         // use 'objRG' to rotate an object
20     protected double scale;                 // use 'scale' to define scaling
21     protected Vector3f post;                // use 'post' to specify location
22     protected Shape3D obj_shape;
23     protected static String obj_name; // For FanBlades and Guard. Setting appearance for multiple parts of
24     protected String texture_name; // Filename for texture string
25
26     protected float x, y, z; // Dimension for square shape
27

```

3.4 Software testing and operation

The main way we tested the functionalities of program was through user interaction. We tested collision detection by having the user run into object and see if the collision detection worked. We tested the book interaction and the door interaction by clicking on these objects at different angles. We also added print message that helped us debug. Some of the debug print messages included: outputting the location of the character when collision detection occurred and whether book swapping was successful. We would then have other people test our program to see if there any bugs that we missed. This test procedure was most effective because with games, there's not a given set of inputs that you can logically determine will cause an error. You need to try your best to cause an error to find them. Like the collision detection, and intentionally trying to go through the objects

4 Discussions and Recommendations

7.1 Discussions about the team projects

The project was a success. We were able to implement all the requirements set by the product owner. We also were able to come up with a good design for Leddy Library. We struggled at first with setting up Azure. Then we had a lot of problems setting up the git

repository. The IDE used by most team members was Eclipse so we learned how to set up a git repository in eclipse. We then had to learn how to use git on the command line. Once everything was set up, the team was able to utilize git to its fullest. Some of the issues that our team member ran into were ineffective collision detection, light flickering not working, game mechanics not working, slow program due to too many objects and much more. We asked the professor for help whenever we can and we learned a lot about game development and how to work together as a team.

There were some things that we couldn't do in java3D. Like predictive collision detection. However, we found work arounds that were acceptable. Also defining large scene graphs in java3D is quite complex and something like Unity would have made the scene creation job much easier. Also we had an issue initially with the eclipse objects. They objects wouldn't render in eclipse because of some code that wasn't recognized by Eclipse. Once we removed those lines of code, the objects were able to be rendered.

7.2 Recommendations for better practices

We did not follow all the Scrum practices accurately at first. Most of the team forgot to move the tasks to 'done' so that we can see an updated burndown chart. We should've talked more about the burndown chart.

We were not implementing design patterns when designing the classes. Once we learned about these design patterns, it led to improved object-oriented design. Now that we have a better understanding of what a sprint should be, we can plan it out better. Better time management and productivity would have given us more time to scale the project up. Ideally in the same time frame, we'd be able to create three or four rooms.

Finally, when we do another project, we will work with the same IDEs. We had an issue with IDE compatibility. One team member was using IntelliJ which has a different project structure than eclipse. So, whenever we tried to merge, we'd have to manually resolve conflicts.

Conclusion

This project successfully created a haunted virtual reality experience based on Leddy Library's second floor. Interactivity is provided through integrated mini-games, and immersion is enhanced via smooth player movement, collision detection, and atmospheric sound and lighting effects. The team utilized Scrum and principles from the Unified Process to practice agile software development methodologies. The project

proved to be a rewarding and valuable learning experience in game development, teamwork, and applying theoretical concepts in practice.

Appendix A List of Backlog Items and Tasks with Brief Descriptions

Interactive books (Bookshelf puzzle): User needs to be able to select a book. This will be done using their mouse. They should see a pointer in the game that represents their mouse cursor.

Tasks

- Sound effect when selecting book (Bookshelf Puzzle): Users should hear a sound when they select a book from the interactive bookshelf with their cursor.
- Movement of books (Bookshelf Game): After selecting a book from the interactive bookshelf, the user can select an empty space in the bookshelf. This will cause the book to move to that space and create an empty space where the book used to be. After movement, book and spaces remain interactive.

Movement of User: User needs to move around the library.

Tasks

- User Movement: User needs to move around the library using the arrow keys.
- User perspective: User needs to see 360

Lighting, sound, atmosphere: User needs to be able to see the library and all the objects. Includes shadows, fog and flickering lights which creates a spooky atmosphere.

Tasks:

Creating light: User needs to be able to see the library and all the objects. Includes flickering lights.

Creating the library: Users need to see a 3D virtualized version of Leddy library.

Tasks

- Arranging the objects: User should see a virtualized version of Leddy library. Floors, bookshelves that contain books, ceiling and walls. These objects should be defined within the library boundary.
- Collision detection: User needs to set up the library boundary. Define where players can move. Collision detection should be defined for all objects (walls, bookshelves, etc)

Sound Design: User should hear background sound and interactive sound

Tasks:

- Sound when books are clicked: The user should hear a sound whenever a book is clicked
- Background sound: The user should hear an ominous background sound in the game

Blender Objects: User should see familiar objects from Leddy library

Tasks:

- Create the cubicle object: Users should see a cubicle object that looks the same as the seats in Leddy library
- Create pillar object: The pillar object for the walls
- Door object: Create the door object for the entrance into Leddy library 2nd floor
- Handle object: Create the handle object for the doors
- Ghost object: Create the ghost object for the ghost that'll block the door

Book games: User should be able to play some games with the books in the bookshelf to escape the library.

Tasks:


- Swapping books logic: When a user clicks on two books, they should swap positions
- Book games: The user should be able to play a different game whenever they click on a book in the bookshelf

Adding details to the library: The objects in the game should look similar to how they look in real life

Tasks:

- Texture mapping: The objects should have similar textures to the textures of the objects in Leddy Library.

Appendix B Signatures for Contribution Table (with table attached in excel file)

Sharmarke Kedie _____ 

Dignesh Solanki _____ 


Rohan Limbachia _____ 

Nasir Giwa _____ 


Khalid Al Fattouhi Al Jundi _____ 

Appendix C Signatures for Participation Table (with table attached in excel file)

Sharmarke Kedie _____ 

Dignesh Solanki _____ 

Rohan Limbachia _____ 

Nasir Giwa _____ 

Khalid Al Fattouhi Al Jundi _____ 