Table 1: Revision History

Date	Developer(s)	Change
September 28, 2018	All members	Pushed Rev0 to repo
December 3, 2018	All members	Changes made for Rev1

SE 3XA3: Development Plan
BlockBuilder

Team 28, OAC
Andrew Lucentini, lucenta
Owen McNeil, mcneilo
Christopher DiBussolo dibussoc

This document describes the necessary components required to pursue BlockBuilder.

1 Team Meeting Plan

Mandatory team meetings will take place every **Tuesday at ~~4:30pm~~ 7:30pm** and every **Thursday at ~~9:30am~~ 5:30pm**. These mandatory meetings will take place at McMaster University ~~in ITB 236~~ on the main level of **Thode Library**, and will continue until **December 1, 2018**. Additional meetings will be held at the discretion of the team. If additional meeting must be held, they will be held **Friday's and/or Saturday's at 11:30am**. The meeting roles for each team member are described below:

Owen McNeil: Meeting facilitator
Andrew Lucentini: Meeting coordinator and leader
Chrisopher Dibussolo: Meeting recorder and time keeper

Meeting agendas will be made for each meeting from a template provided by the meeting ~~manager~~ recorder. A template of these meeting minutes can be found here. ~~in the ReferenceMaterial folder in the root directory of the team repository.~~

2 Team Communication Plan

The team will communicate via Discord or Whatsapp, two online communication tools. These tools will be used to discuss the project and organize team meetings. In the case of an emergency, team members will contact each other via cellular device. The team members must guarantee a 24 hour ~~reponse~~ response time, or let the team know if they will be unresponsive for a duration more than 24 hours. Team members must communicate to the group members

when they perform a push to the git repo. Any exchanging of files will be done via GitLab.

3 Team Member Roles

The following table lists the roles for each team member in the group.

Name	Roles
Andrew Lucentini	Software Developer Program Tester Meeting Coordinator Task Submitter
Christopher DiBussolo	Project Manager Software Developer Program Debugger
Owen McNeil	Project Developer Program Debugger Software Developer UI Designer

4 Git Workflow Plan

The development team will follow a simplified version of the work flow described in "A successful Git branching model" [1]. The project code will be hosted in one centralized repository, via GitLab, with a branch structure consisting of the master branch and multiple feature branches that correspond to the project milestones. Given the nature and relatively small scale of the project, additional complications such as release, hotfix and develop branches have been deemed unnecessary.

The main advantage of utilizing feature branches is the ability to easily track, merge and modify any project milestone individually, without risk of unwanted changes to other project components. Features will be developed in their own branches, tested if needed, then merged with the master branch and tagged. Feature modification is done by simply modifying the corresponding feature branch and re-merging.

5 Proof of Concept Demonstration Plan

5.1 Technical Feasibility

Given that BlockBuilder has already been implemented in python using the pyglet graphics library, we know that making the game is at least feasible. However, the challenge is whether or not the redesign will meet the requirements. In addition, no member of the team is familiar with pyglet. This can cause delays when attempting to re-implement the software. In addition, there are bugs in the current implementation of BlockBuilder, so it may be difficult to rule those out. It is possible that we may have to use other graphics libraries to meet our requirements.

5.2 Possible Stumbling Blocks

As mentioned above, the team members are not familiar with the graphics library pyglet, which is used in the original implementation of BlockBuilder. The greatest stumbling block will most likely be trying to meet the requirements with the re-implementation of BlockBuilder whilst the team members learn how to use pyglet. Additionally, the original project is kept in one complete main file, reducing readability and making error catching and debugging much more difficult should something go wrong and the team needs to reference the original.

5.3 What Python can or can't provide?

BlockBuilder will be programmed in python. Luckily, python is more than capable for designing relatively simple 2D and 3D games. Given that the team will be building a simpler version of an already simple game, relative to any AAA game, the platform should provide the team with substantial tools to get the job done. Both pyglet and pygame are viable python multimedia libraries for the purpose of this project. In addition, both of these libraries can easily be installed on any computer.

5.4 Scope and Level of Customization for this Project?

The level of customization for this project will entirely depend on how well the existing project meets the requirements the team sets out for their re-implementation. Pyglet and Pygame should both be more than capable of handling any further customization the team may require. The scope of this project should be similar to the original, as the entire game can be completely designed using only python.

6 Technology

BlockBuilder will be written in the python programming language. In addition, the pyglet library is required to run the program. Basic text editors will be used during the development of the software. These text editors will be Sublime Text, Atom, and Visual Studio Code. BlockBuilder will be tested via the pytest framework. This framework makes it easy to write small tests, and supports complex functional testing for applications. Document generation will be generated via Doxygen. This software will generate online and offline documentation.

7 Coding Style

BlockBuilder will follow the coding style of PEP 8. A detailed breakdown of this coding style can be found at the link:
https://www.python.org/dev/peps/pep-0008/?

8 Project Schedule

~~The project schedule can be found in the ProjectSchedule folder in the root directory of the teams repository.~~ The project schedule can be found here.

9 Project Review

Overall, the process of implementing BlockBuilder and the respective documentation was successful considering that all members had little to no background experience in 3D game development. Most of the documentation and coding went according to plan as the team implemented what was required in reasonable amounts of time. One of the most challenging aspects of the project was implementing the code for the main functionality of the program. Specifically, our team had issues implementing collision detection, which is a necessary component to ensure that the player collides with blocks rather than traveling through them. This process involved lots of research and studying of the original open source code to understand how collision detection works. In addition, the team found it difficult to implement the MIS for BlockBuilder. The nature of 3D games make it difficult to mathematically describe modules and access routines. The team often found that many of the access routines had to be explained in English, which is not desired as ambiguity can be introduced.

In the future, the team would modify the development plan by assigning more specific roles to team members. For example, instead of assigning the role of "Program Tester" to Andrew, the team would assign Andrew the role of "Unit Program Tester", and assign Christopher the role of "Manual Program Tester" and "Code Inspector". Assigning more specific roles would have helped organize the work flow of the project and allow individuals to focus on smaller tasks one at a time. The team would also reduce the frequency of mandatory of team

meetings. The team often found it difficult to meet up at the requested times. It would have been more feasible to have one required meeting every two weeks, then have occasional, spontaneous, and less formal meetings throughout the week. Overall, the team worked well together and enjoyed the challenge that the project had to offer regardless of the fact that it was difficult at times to learn about 3D gaming.

References

[1] Vincent Driessen: A successful Git branching model, `https://nvie.com/posts/a-successful-git-branching-model/`