SE 3XA3: Module Guide
BlockBuilder

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

December 5, 2018

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	Version	Notes
11/9/2018	1.0	Initial Creation of Modular Guide based on MIS version 1.0
12/5/2018	1.1	Rev1 documentation finished based on feedback

1 Introduction

1.1 Overview

BlockBuilder is a python re-implementation of the famous open world sandbox game Minecraft, initially developed by Mojang and Mark Perssun. The goal of the BlockBuilder project is to explore the capabilities of the python multimedia library Pyglet by making a stripped down (excluding detailed game mechanics) version of Minecraft free to all users of any age, focusing on world generation and player interaction within the world. The BlockBuilder design team has added multiple requirements that they believe the open source fails to meet, as seen in the SRS. The team's primary focus for the project is on the modularization of the program, particularly focusing on maintainability, to make it as efficient as possible when attempting to implement new features in the future. Similar to the goal of original open source, the team wishes to alter the design of the project in a way that makes the open source a learning tool for others who wish to gain experience in 3D rendering and game design, since as the open source stands, the team believes it is very unorganized and its potential as a learning tool is certainly lacking. For a breakdown of the projects new design and the modules the team has decided to break it down into, see the Module Interface Specification.

1.2 Scope

The scope of the project is completely based around the use of the multimedia python library Pyglet and implementing the basics of the Minecraft game world. The project will not delve into the advanced features of the game such as inventory systems and crafting, as the goal of the project was to explore the team's ability to work with 3D modelling and game world generation as well as delivering an entertaining game that allows the user to alter the world as they please. Pyglet has been chosen as the sole development tool for this project.

1.3 Module Guide Purpose

When developing a 3 dimensional game world, that are many components that must come together to form a working whole. The purpose of this Module Guide is to outline how the team has chosen to break up the required components that comprise BlockBuilder, as well as how these components interact with each other. This Module guide is intended for the use of the the design team to check for consistencies as the code and other documentation is developed as well as for any developer or maintainer that may need to make changes to the project in the future. This Module Guide is intended to provide developers with a means of understanding how the projects components interact as well as providing them with a means of verifying the systems integrity.

2 Anticipated and Unlikely Changes

This section outlines changes the design team has considered making, and breaks them down into feasible changes the team anticipates implementing, as well as changes that are good in theory, but are unlikely to improve the quality of the project in reality.

2.1 Anticipated Changes

AC1: The hardware on which the game is ran.

AC2: The available game textures.

AC3: The window resolution, framerate and default size.

AC4: The default player input settings.

AC5: Objects the player is able to interact with.

2.2 Unlikely Changes

UC1: Input devices. It is possible that the team may wish to change the input to a generic game console controller as opposed to a keyboard and mouse. However, given the scope of the project, it is unlikely this change will occur.

UC2: Creating a more general method of storing the game world (i.e loading chunks) to allow a larger world playing field. Changing this aspect of the design would force many changes on components that interact with it, making this change unlikely.

3 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 2. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

M1: Window

M2: World

M3: Block

M4: Constants

M5: Function

Level 1	Level 2
Hardware-Hiding Modules	Window
Behaviour-Hiding Modules	World Constants
Software Decision Modules	Block Function

Table 2: Module Hierarchy

4 Connection Between Requirements and Design

This program was designed to fulfill all requirements stated in the SRS Document. BlockBuilder is broken into several loosely coupled modules in order to satisfy all functional requirements, such as supporting complex terrain generation and utilizing third party frameworks for 3D support. The explicit connection between requirements and modules is listed in Table 3.

5 Module Decomposition

The purpose of module decomposition is to improve the principle information hiding. The idea behind information hiding is to reduce program complexity while improving readability and program maintainability. All modules contain secrets and services. The secrets of a module are things kept hidden in that module to other modules. The services of a module specify what the module will provide to other modules without the need to understand its implementation. For each module listed in the module hierarchy, the secrets and services kept and provided by the respective modules are listed below.

5.1 Hardware Hiding Modules (M1)

Secrets: The algorithms used to act as the virtual input and output hardware such as the keyboard and mouse and the screen.

Services: Serves as a virtual hardware for the keyboard and mouse as well as the screen, used by the operating system to take in input from the user and output the in-game results to the screen.

Implemented By: OS

5.2 Behaviour-Hiding Module

Secrets: Required behaviours of the player and Game World.

Services: Includes programs to ensure proper externally visible player and world behaviour as specified in the SRS document. The programs in these modules will have to be altered alongside any changes in the requirements. These modules act as a means of communication between the software decisions module and the hardware-hiding module.

Implemented By: Block Builder

5.2.1 Game World Operations Module (M2)

Secrets: The game world operations.

Services: Details the behaviour of functions that manipulate the game world based on input from the Window module.

Implemented By: Block Builder

5.2.2 Game World Constants Module (M4)

Secrets: The game world constants.

Services: Details the constants used by the World and Window modules that must remain consistent throughout the game world. Identifies motion logic constants, texture types, etc.

Implemented By: Block Builder

5.3 Software Decision Modules

Secrets: Algorithms used for various OpenGL and 3D world generation operations.

Services: Includes data structure and algorithms used for generating the game world that do not provide any direct interaction with the player.

Implemented By: –

5.3.1 Software Decision Module M3

Secrets: Algorithms used specifically the generation, detection and interaction of blocks in the world.

Services: Includes data structure and algorithms used for generating, detecting and interacting with blocks in world.

Implemented By: Block Builder

5.3.2 Software Decision Module M5

Secrets: Algorithms used for world grouping.

Services: Includes data structure and algorithms used for separating the world into more manageable parts for the computer.

Implemented By: Block Builder

6 Traceability Matrix

This section shows two traceability matrices: between the modules and the requirements and between the modules and the anticipated changes.

Req.	Modules
~~R1~~	~~M4, M3, M2~~
~~R2~~	~~M5, M3~~
~~R3~~	~~M2~~
~~R4~~ FR4	M1, M2, M3, M4, M5
~~R5~~ FR5	M3, M2, M1
~~R6~~ FR6	M1, M5, M2
FR7	M1, M5, M2
FR8	M1, M5, M2
FR9	M1, M5, M2
FR10	M1, M5, M2
FR11	M1, M5, M2
FR12	M1, M5, M2
FR13	M1, M5, M2
FR14	M1, M5, M2
FR15	M1, M5, M2
FR16	M1, M5, M2, M3, M4
FR17	M1, M5, M2, M3, M4
FR18	M1, M5, M2, M3, M4
FR19	M1, M5, M2, M3, M4
FR20	M1, M5, M2, M3, M4
FR21	M1, M5, M2, M3, M4
FR22	M1, M5, M2, M3, M4

Table 3: Trace Between Requirements and Modules

AC	Modules
AC1	M1
AC2	M4
AC3	M1
AC4	M5
AC5	M2

Table 4: Trace Between Anticipated Changes and Modules

7 Use Hierarchy Between Modules

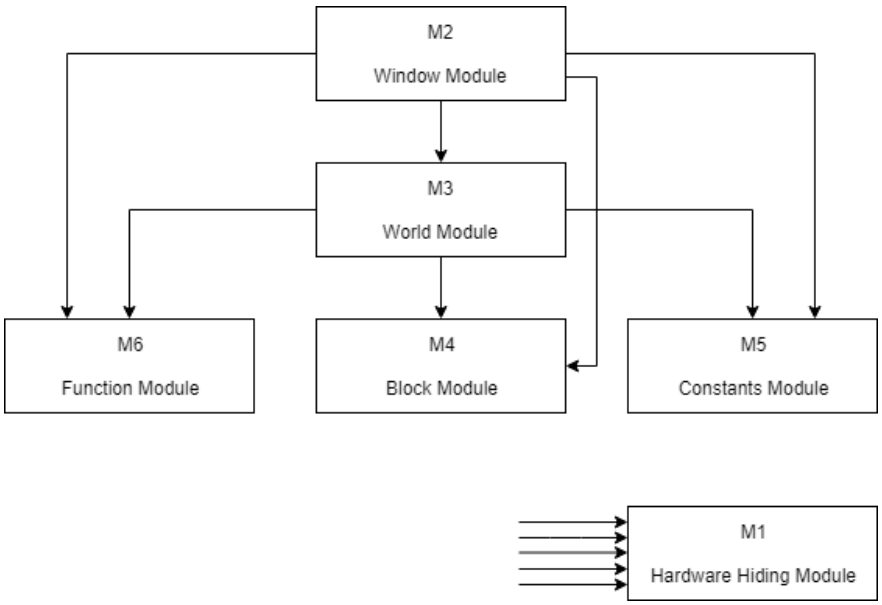

Figure 1: Use hierarchy among modules