1 INTRODUCTION

This maintenance guide provides maintenance personnel with the information necessary to maintain the system effectively. It provides the definition of the regular activities essential to the support and maintenance of program modules.

1.1 Project Reference

The project consists of implementing and designing a game. A team of four students implemented the game in an iterative way using Scrum methodology. The topic of project is Jungle Hunt, a classical game that has been implemented during four sprints.

2 SYSTEM DESCRIPTION

2.1 System Application

As mentioned earlier our system is a game which we implemented it making use of Unity platform. This game environment includes a player which is present in all the scenes. The user controls the player with keyboard buttons and tries to pass each level by reaching to a specific place in the corresponding scene.
While classic Object-Oriented Programming (OOP) can be and is used in Unity, the Unity workflow highly builds around the structure of components. So, most of functionalities for different components of the game are implemented within different classes. Some other functionalities are not deployed in classes and are just implemented by Unity's graphical interface. Each class is delivering a different functionality as shown on the following graph.

Class **GameManager**	“heart of game”, controls whole gameplay
Class **StoryTeller**	responsible for cutscenes
Class **Singleton**	Instance of GameManager
Class **SaveManager**	saves information about the player between game sessions
Class **CameraSystem**	responsible for displaying the right camera view to the user.
Class **Health**	responsible for displaying number of remaining health.
Class **Points**	responsible for displaying number of collected points
Class **PlayerMovement**	responsible for player movement and triggering right animations for every movement.
Class **VineMotor**	responsible for vines’ movement.
Class **PlayerAutoSwim**	responsible for player auto-movement under water.
Class **Bubble**	responsible for movement and physics of bubbles.
Class **BubbleSpawner**	responsible for generating new bubbles ahead of the player.
Class **CrocSwim**	responsible for crocodiles’ movements.
Class **PlayerAutoMove**	responsible for player auto-movement and triggering right animations for every movement.
Class **Boulder**	responsible for movement and rotation of boulders.
Class **BoulderSpawner**	responsible for generating new boulders ahead of the player.
Class **CannibalWalk**	responsible for cannibals’ movements.

2.2 System Organization

Unity works by splitting the levels to different “scenes”, which all have their unique assets. This means that the levels are independent of each other and will not break each other’s functionalities.

3 SUPPORT ENVIRONMENT

3.1 Support Software

Our game is made with Unity Version 2017.3.1f1 (64bit). For maintenance naturally same version is preferred for compatibility. You also need code editor for example Visual Studio 2015 to view code views. Image viewer for example Ifranview is needed for viewing graphics.

3.2 Personnel

Person in charge of maintenance should have experience with Unity. Programming knowledge of C# is also needed. For changing graphics some kind of photo editing experience might be needed.

4 SYSTEM MAINTENANCE PROCEDURES

This section contains information on the procedures necessary for programmers to maintain the software.

4.1 Conventions

The project uses common C# coding standards, as stated by Microsoft. [1]

5 Sources

[1]https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions