Landslider: A Depth Detection Game

Requirement Specification Document

Team 5
Michael Fadem
Cameron Chinena
Christopher Gonzales
Eduardo Lash
Jeremy Keys

Table of Contents

1 Introduction

The product that is being designed and built is a video game, in which a player will manipulate different rows of land tiles, 3D squares with varying terrains, that are at varying depths(distances) from the player's perspective and adjust them by pulling them closer or pushing them away in such a way as to allow a non-playable character to traverse the level. These actions will be accomplished by tracking the user's hands and fingers and recognizing the movements and gestures as gameplay input. This game will be built utilizing the Unity game engine and Intel RealSense F200 camera. It will also have research potential if a researcher wants to analyze how depth affects gameplay.

1.1 Purpose of Product

The purpose of this product is to demonstrate and experiment with how players might use gesture inputs and depth elements to control objects in a virtual space. The problem that is being explored with this product is how players might interact with the game's environmental aspects at different depths (distances) from the player's perspective, specifically by using the RealSense camera. If a player is using a controller, then the depths aspect are not readily apparent to the player, whereas if the user has to physically move their arms further away or closer to themselves, then they will be better connected to the gameplay and virtual space.

1.2 Scope of Product

What is planned to be accomplished with this product is to create a game that allows for the user to use hand, finger and depth tracking to manipulate squares that are at different depths from the player's perspective. These squares will contain different types of land terrain such as volcanoes, spikes, snowmen, forests, or rivers that will impede the characters movement through the level. The player using their hands will move rows of these tiles back and forth so that the character will be able to traverse the level. The game will feature prefab tiles that a researcher can use to assemble a level that will satisfy the researchers desired test parameters within the game environment. Lastly, for research purposes the game will record data such as the RealSense camera output, score, length of session, and land mass placement so that a researcher can see how a session was affected by different depths and terrains. Since this game is designed for more experimental purposes the game will not feature some typical elements commonly found in video games such as leaderboards, time limits, or difficulty levels.

1.3 Acronyms, Abbreviations, Definitions

- ❏ Gesture-based human-computer interaction - This term has emerged in the past decade to refer to interactions between humans and computers that rely on gestures, especially hand and facial gestures, in order to carry out actions on the computer.

- ❏ Intel RealSense F200 3D Camera - Hardware developed by Intel that uses various sensing technologies to achieve depth perception tracking, hand tracking, and 3D scanning. This specific model is the standard model of the RealSense cameras which is to used as a user facing camera. The player will interact with this camera during gameplay.

- Unity Game Engine - Free to use video game creation engine, integratable with Intel RealSense cameras
- SDK - Software Development Kit
- XML - Metalanguage which allows users to define their own customized markup languages
- NPC - Non-Playable Character, this will be the character that is running indefinitely in the game.

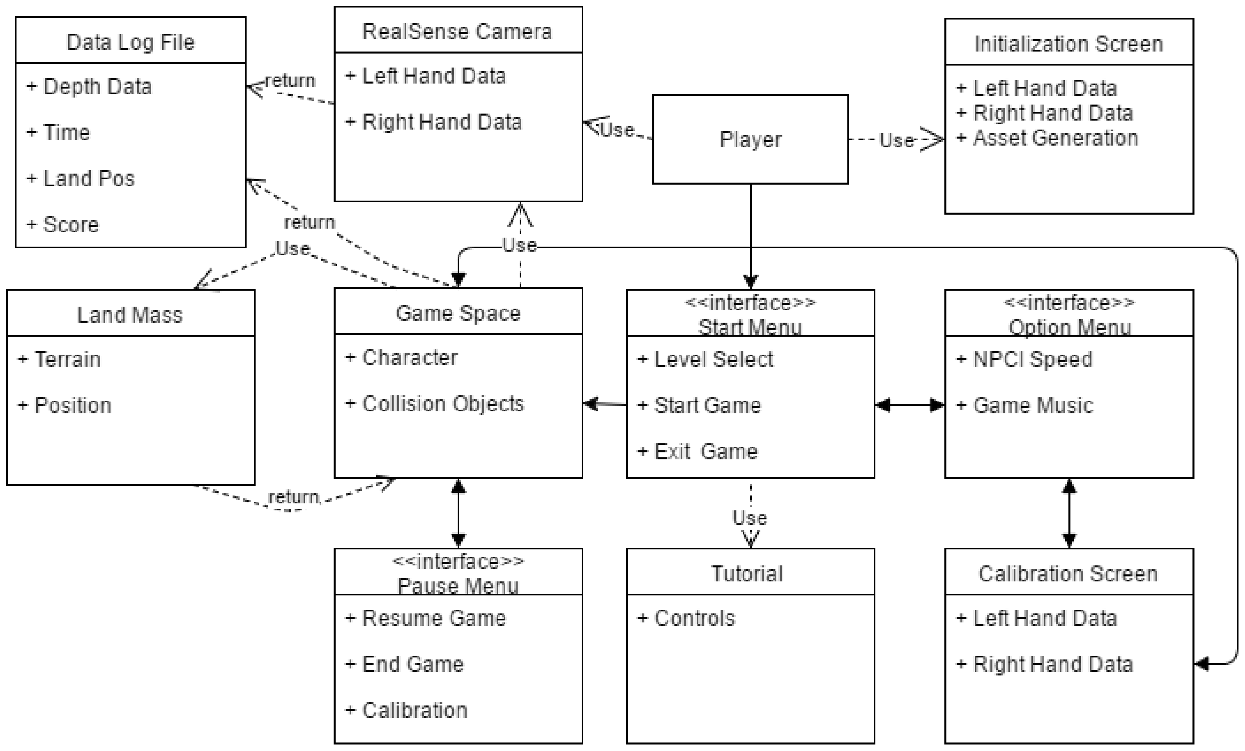
1.4 References

Unity Game Engine Documentation
https://docs.unity3d.com/Manual/index.html
C# Documentation
https://msdn.microsoft.com/en-us/library/gg145045(v=vs.110).aspx
RealSense Documentation
https://software.intel.com/en-us/intel-RealSense-sdk
https://software.intel.com/sites/landingpage/RealSense/camera-sdk/v1.1/documentation/html/index.html?doc_devguide_introduction.html

2 General Description of Product

This camera will detect a player's hands, gestures, depth, and distance and this will correspond to where the player's hand is displayed in the game space. With this data, the game can detect when the player is in the right position to grab a land mass and push/pull it, resulting in a clear path for the NPC on screen.

2.1 Domain Model



2.2 Context of Product

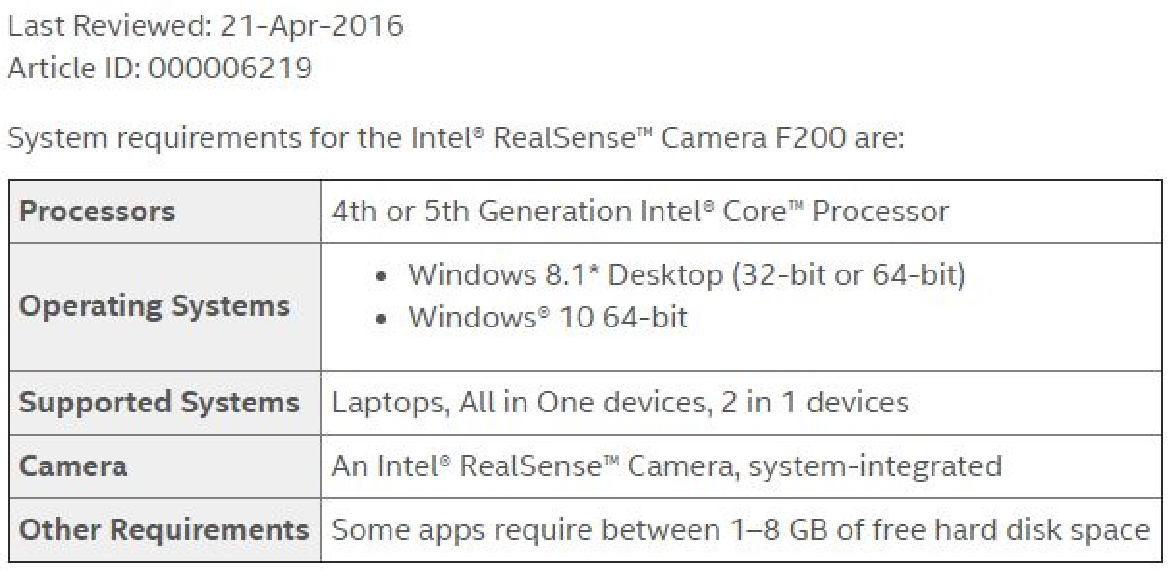
The minimum computing environment requirements for the camera are the Intel RealSense drivers and SDK, 4th-Gen Intel core processor, at least 4GB of memory, and a 3.0 USB port. The minimum requirements for Unity are Windows 7+, graphics card with DirectX 9 or 11, Unity SDK, C#, and the Visual Studio IDE. The game will be primarily used in a lab setting due to its research opportunities, however it can be run in a home environment provided the computer used meets the minimum requirements. In the settings where the game will be being played it will be good to have a wide playing space to ensure that the player isn't restricted in their movements.

2.3 User Characteristics

What is expected from the average player is that they have some familiarity with the concept of gesture based controls due to the proliferation of devices such as the Microsoft Kinect and Playstation Eye. The assumption of familiarity with the concept leads into the assumption that the the average player will have used a system similar to the RealSense. In the event that the user has not used a such a system a tutorial will be provided so that the player will have an understanding of game mechanics.

2.4 Constraints

The Intel RealSense F200 camera have the following system requirements (image from Intel's RealSense documentation):



Last Reviewed: 21-Apr-2016
Article ID: 000006219

System requirements for the Intel® RealSense™ Camera F200 are:

Processors	4th or 5th Generation Intel® Core™ Processor
Operating Systems	• Windows 8.1* Desktop (32-bit or 64-bit) • Windows® 10 64-bit
Supported Systems	Laptops, All in One devices, 2 in 1 devices
Camera	An Intel® RealSense™ Camera, system-integrated
Other Requirements	Some apps require between 1–8 GB of free hard disk space

The external constraints that will impact the game are the system requirements for the RealSense camera and the Unity game engine. The computing demands for these external systems were outlined in section 2.2 Context of Product.

2.5 Assumptions and Dependencies

The Unity Engine works with DirectX 9 and DirectX 11, which is a potential Windows dependency. However, Unity applications will fall back on OpenGL if DirectX is not accessible. Therefore, the product could package DirectX with a Windows version of the game, or the assumption could be made that the machine has DirectX and, if not, just allow the Unity application to automatically fall back to OpenGL. Unity's ability to use different graphical APIs is part of the reason why a single Unity application can be built for multiple platforms, including Windows and Linux. DirectX can thus be considered an "optional dependency."

3 Specific Requirements

This section will describe what's expected of the project, as well as go over the various features it will include.

3.1 External Interface Requirements

Primarily the player will interface with the hand tracking and depth detection through the RealSense camera. More details on the specifics of this interface will be detailed below.

3.1.1 User Interfaces

The user interface will utilize the hand/gesture-based tracking offered by the RealSense camera to allow users to interact with the menus using their hands. To complement the RealSense enabled menu system, the game will also include mouse enabled buttons as a substitute for the main method of navigation.

3.1.2 Hardware Interfaces

The key hardware component that will be used to interact with the game is the Intel RealSense F200 3D Camera. The RealSense camera have advanced scanning technologies that allow for gesture recognition, 3D scanning, and hand/finger tracking. This specific camera is designed to be the user facing camera which entails that is is built for close range detection. Based off of Intel's specification for the F200 camera it’s make raw data range is between seven and thirty one inches.

3.1.3 Software Interfaces

The program will be entirely developed in the Unity Game Engine, interfaced with the Intel RealSense SDK.

3.2 Functional Requirements

The following sections will describe in detail the software components and what will be needed from each component to be considered complete and fully functional.

3.2.1 Options Menu

The options menu will be accessible from the pause menu and the main menu. This menu will have two options, such as music audio, and NPC speed.

3.2.2 Main Menu

The main menu will contain multiple buttons that the user can select. These selections will be start game, calibration menu, options menu, and exiting the game.

3.2.3 Pause Menu

The pause menu will be accessible with a specific gesture or mouse click. It will pause the game preventing the character from moving. The pause menu will contain the following options: a resume game button, main menu button, and options button.

3.2.4 Calibration Screen

At any moment the user can access the calibration screen if the RealSense camera isn’t registering the hand data correctly. The calibration screen will enable the user to see what the camera is registering to ensure they are in the correct location. Also it will allow the user to test if the gesture recognition is functioning correctly.

3.2.5 Game Screen

The game screen will feature a basic user interface showing the current health of the NPC the NPC itself, the specific level that the user is currently on, and the level itself. The health of the character will be represented with a health bar in the corner of the screen, when the user fails to avoid an obstacle, an amount of health will be removed.

3.2.6 Data Log File

In order for the a researcher to analyze a session of the game the system will output a data log file, .csv or .txt, that will contain data such as RealSense camera data (depth, x,y, and z axis), score, length of session, and land mass placement so that a researcher can see how a session was affected by different depths and terrains

3.3 Performance Requirements

To ensure gameplay is as smooth as possible for the player, the system will ensure that the player is satisfying the optimal conditions to play the game with the use of calibration and tracking. Also the depth and gesture detection should be quick and responsive with minimal delay to ensure that aspect of the game doesn’t take away from the experience. Lastly, it is paramount to ensure that the gameplay itself does not result in crashes or errors that will cause the game to become nonfunctional, any errors should be restricted to camera detection errors which can be corrected by the calibration screen.

3.4 Design Constraints

Our project is restricted to using the RealSense camera, so we must design our project to use features that the camera provides, such as tracking basic human gestures. That means the project is also limited to basic human movement, so the controls of the game need to not be too complex nor impossible for the human body to perform.

3.5 Quality Requirements

The user will be given a tutorial level to explain the basic controls of the game, and how to progress through the game. The hand tracking should be responsive and accurate to ensure that the user will not make an error or lose the game because of poor tracking. If hand tracking data is lost, the user has moved out of the range of what the camera can detect, the user should pause the game, and proceed to the calibration screen that gives feedback as to where the user should stand to detect movement again. Since there is research potential for this game it is essential that the NPC’s data is never lost so that it can be exported upon game completion.