IQRA UNIVERSITY IU

Learning Environment using Leap Motion

Submitted To: Sir Jamal Haider Zaidi

Submitted By: Syed Hunain Raza Zaidi - 15063
Abdul Hadi - 15598

Acknowledgement

The completion of this project could not have been possible without the participation & assistance of many people. Their contributions are sincerely appreciated. However, the group would like to express deep indebtedness particularly to the following:

Sir Jamal Haider Zaidi for his endless support, kind & understanding spirit during this project.

To all relatives, friends & others who supported, either morally, financially & physically.

Above all, to the Great Almighty, the author of wisdom & knowledge.

We thank you.

Table of Contents

Overview

Our aim is to develop a game which would help students learn real life physics through a 3D environment, which would demonstrate physics principles and factors affecting those principles. Through this game we intend to help teachers to clear concept of students which can be made cleared in a physics physical class.

Problem Statement

Infectious disease pandemic like COVID-19 spreads easily through social contact and presents a challenge to public health, hence resulting in less interactions.

Covid-19 has caused closure of school, colleges & universities around the world making it hard for students to gain knowledge and skill production. In such situation the whole world is opting different ways to provide better & quality education.

Our problem is to build a game which will simulate certain principles of Physics. This game will help students to understand real-life examples through a game.

Technical Background

Unity

Unity is a cross-platform game engine initially released by **Unity Technologies**, in 2005. The focus of Unity lies in the development of both 2D and 3D games and interactive content. Unity now supports over **20** different target platforms for deploying, while it's most popular platforms are the PC, Android and iOS systems.

Unity is a 2D/3D engine and framework that gives you a system for designing game or app scenes for 2D/3D. By saying games and apps means that there are not just games, but training simulators, first-responder applications, and other business-focused applications developed with Unity that need to interact with 2D/3D space. Unity allows you to interact with them via not only code, but also visual components, and export them to every major mobile platform and a whole lot more for free (There's also a pro version which can be bought). Unity supports all major 3D applications and many audio formats, and even understands the Photoshop .psd format so you can just drop a .psd file into a Unity project. Unity allows you to import and assemble assets, write code to interact with your objects, create or import animations for use with an advanced animation system, and much more.

Perhaps the most powerful part of Unity is the Unity Asset Store, one of the best asset marketplace in the gaming market. In it you can find all of your game component needs, such as artwork, 3D models, animation files for your 3D models, audio effects and full tracks, plug-ins including those like the Multiplatform toolkit that can help with multiple platform support, visual scripting systems such as PlayMaker and Behave, advanced shaders, textures, particle effects, and more. The Unity interface is fully scriptable, allowing many third-party plug-ins to integrate right into the Unity GUI. Most, of the professional game developers use a number of packages from the asset store.

C#

C# is a high-level class-based component-oriented general-purpose programming language built as an extension of C. C# was developed in Microsoft around 2000 by Danish software engineer Anders Hejlsberg and his team as a part of the .NET architecture. At the moment, C# is on its 9th release.

Currently, C# is the dominant scripting language for the Unity (a.k.a. Unity3D) platform as both its alternatives, Boo and UnityScript, were deprecated back in 2017. UnityScript, once a popular version of JavaScript modified for the Unity, was deprecated since only 3.6% of projects heavily utilized the language, albeit the time spent on support was substantial.

Coding in C# allows you to develop unique custom solutions that are not well-documented yet, optimize your applications for better performance, integrate external code to save time on development, and have maximum control over how your application works.

Unity Programming Language (C#) Vs. Unreal Programming Language (C++):

For years Unity was considered a default choice for indie or DIY game development whereas Unreal Engine was more appropriate for large-scale projects given its extensive history of being used in AAA titles.

Although both game engines have matured over the years and proved that they can be used to develop projects of varying scales, there are still a couple of factors that make Unreal Engine's C++ environment more suitable for massive or technically demanding projects:

- **C++ code can be faster and more efficient**. Given that C++ allows manual memory management and compiles directly into machine code, large-scale applications can be optimized for maximum performance whereas with Unity's C# such program efficiency is out of reach.
- **Unreal Engine C++ code is open-sourced**. Unlike with Unity where only the managed C# portion of source code was made available to public under a reference-only license (no reuse or modification), Unreal Engine's C++ source code is fully available for access and modification, something most AAA projects require.

Popularity and Community:

According to the TIOBE Index that indicates the general popularity of programming languages, both C+ and C# are in the top 5 most popular with C++ (7.36% rating) being slightly more common than C# (5.14%).

At the same time, C++ is the most commonly used coding language for building game engines to the point that any AA engine is written at least partially using C++.

Game Engines:

C++	C#
CryEngine	Xenko
Lumberyard	Wave
Source Engine	Duality
Unreal Engine	Otter2D
G3D Innovation Engine	
IrrLicht	
Urho3D	

However, the sheer amount of projects developed with Unity make C# one of the most popular programming languages in game development. According to Unity's website, more than 50% of games across PC, mobile, and console were made using the platform. If we are talking XR only, Unity constitutes to about 60% of all the AR/VR content.

Unreal Engine's community had around 7 million developers and designers in 2018, whereas Unity reported 1.5 million active monthly creators in 2020.

It's safe to say that comparing both C++ and C# in terms of popularity in the game development industry will be a close call for years to come. Or in other words: you can't go wrong with either choice.

Learning Curve:

C# language is easier to learn than C++ for several reasons:

- **Easier syntax.** Being a higher-level language, C# syntax is less error-prone than C++ and is relatively easy to learn.
- **The bigger margin of error.** C# handles garbage collection automatically which helps beginner developers to avoid memory leakage and not spend time debugging their code. C++ typically requires a manual approach to memory management with a higher chance of error.
- **Many Unity-specific C# tutorials**. Unity's community is more welcoming towards beginner C# developers. Unity is more often used for DIY and indie projects that are of interest to someone just starting out in game dev.

Garbage Collection:

Garbage collection (GC) refers to the process of collecting or gaining memory back when it's not currently being used by parts of an application.

One of the differences between C# and C++ is that C# code runs on a virtual machine with built-in automatic garbage collection while C++ needs manual memory management.

This language-level difference translates into difference when developing your app in Unity or Unreal Engine. For example, Unity uses modified Mono as its virtual compiler and there's not much Unity developers can do in terms of memory management other than relying on Mono's built-in garbage collection system which is not always optimal.

Unreal Engine 4 also utilizes a garbage collector engineered directly by Epic. However, Unreal Engine developers can avoid using a built-in garbage collector altogether and instead rely on best practices for resource management.

Comparing memory management in C++ and C#, it all comes down to the developers' experience. Experienced C++ developers have a higher degree of control over how their apps use system resources and hardware and thus can achieve exceptional efficiency. At the same time, C++ devs need to be careful in avoiding memory leaks that are notoriously hard to track. Less experienced developers have to rely on built-in solutions albeit having less control over the application's performance.

Summary of C# and C++ Pros and Cons:

C# Pros:	**C++ Pros:**
Easy to learn	Most commonly used in AAA projects and high-tier game engines
More suitable for DIY and indie projects	More control over application resource management and engine's source code
Commonly used in Microsoft applications	
Cheaper development	
C# Cons:	**C++ Cons:**
Less control over application performance	Higher chance of error
Not many applications outside of Unity and Microsoft infrastructure	Takes lots of experience to make the most of its capabilities
	Development requires more experienced employees, hence is more expensive

Blender

Blender is a free and open-source 3D computer graphics software toolset used for creating animated films, visual effects, art, 3D printed models, motion graphics, interactive 3D applications, virtual reality, and computer games. Blender's features include 3D modelling, UV unwrapping, texturing, raster graphics editing, rigging and skinning, fluid and smoke simulation, particle simulation, soft body simulation, sculpting, animating, match moving, rendering, motion graphics, video editing, and compositing.

Features:

Modeling

Modifiers:

Modifiers apply non-destructive effects which can be applied upon rendering or exporting.

Sculpting:

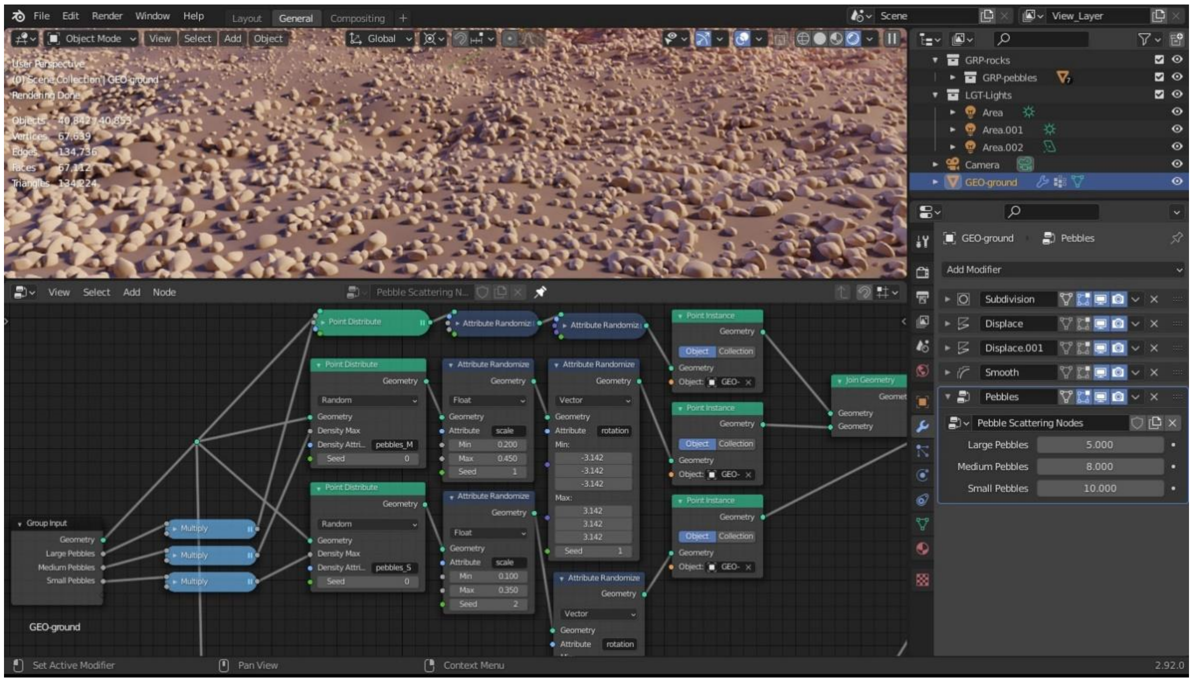
Blender has multi-res digital sculpting, which includes dynamic topology, maps baking, re-meshing, re-symmetrize, and decimation. The latter is used to simplify models for exporting purposes. E.g. to use in a game.

Primitives:

Blender has support for a variety of geometric primitives, including polygon meshes, fast subdivision surface modeling, Bézier curves, NURBS surfaces, metaballs, icospheres, text, and an n-gon modeling system called B-mesh.

Geometric nodes:

Blender's Geometry nodes is a node based system for procedurally and non-destructively creating and manipulating geometry. It was first added to Blender 2.92, which focuses on object scattering and instancing. It takes the form of a modifier, so it can be stacked over different modifiers. The system uses object attributes, which can be modified and overridden with string inputs. Attributes can include Position, Normal and UV maps. All attributes can be viewed in an attribute spreadsheet editor. Geometry Nodes also has the capability of creating primitive mesh such as Cubes, Spheres, Icospheres and Cylinders. In Blender 3.0, support for creating and modifying curves objects will be added to Geometry Nodes. In Blender 3.0, the geometry nodes workflow was completely redesigned with fields in order to make the system more intuitive and work like shader nodes.



Hard surface modelling:

Hard surface modeling is usually used to design hard surfaces such as cars and machines. It is usually done in a non-destructive (using as many modifiers as possible) manner but can be destructive.

Simulation

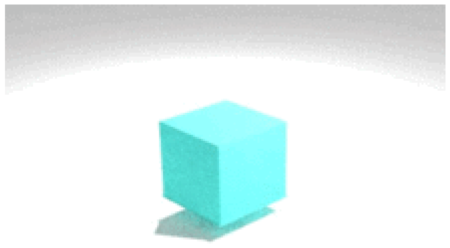
Blender can be used to simulate smoke, rain, dust, cloth, fluids, hair, and rigid bodies.

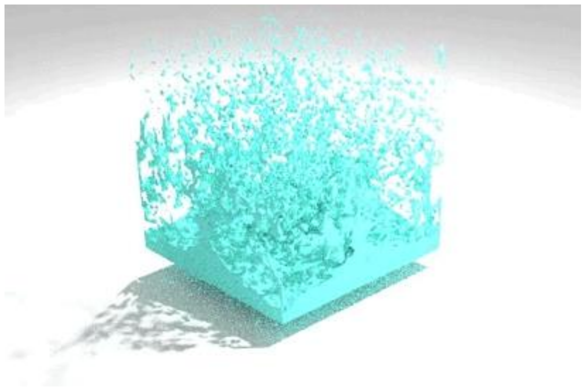
Fluid Simulation:

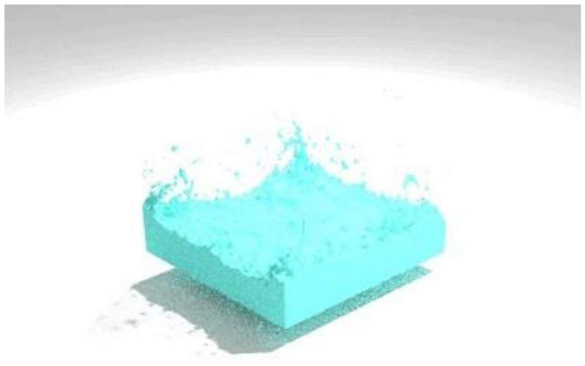
The fluid simulator can be used for simulating liquids, like water hitting a cup. It uses the Lattice Boltzmann methods (LBM) to simulate the fluids and allows for lots of adjusting of the amount of particles and the resolution.

The particle physics fluid simulation creates particles that follow the Smoothed-particle hydrodynamics method. Simulation tools for soft body dynamics including mesh collision detection, LBM fluid dynamics, smoke simulation, Bullet rigid body dynamics, and an ocean generator with waves. A particle system that includes support for particle-based hair. Real-time control during physics simulation and rendering.

In Blender 2.82 a new fluid simulation system called mantaflow was added, replacing the old system. In Blender 2.92 another fluid simulation system called APIC was added. Vortices and more stable calculations are improved in Relation to FLIP system. Improved Mantaflow is the source of the APIC part.







Animation

Key framed animation tools including inverse kinematics, armature (skeletal), hook, curve and lattice-based deformations, shape animations, non-linear animation, constraints, and vertex weighting.

Grease Pencil:

Blender's Grease Pencil tools allow for 2D animation within a full 3D pipeline.

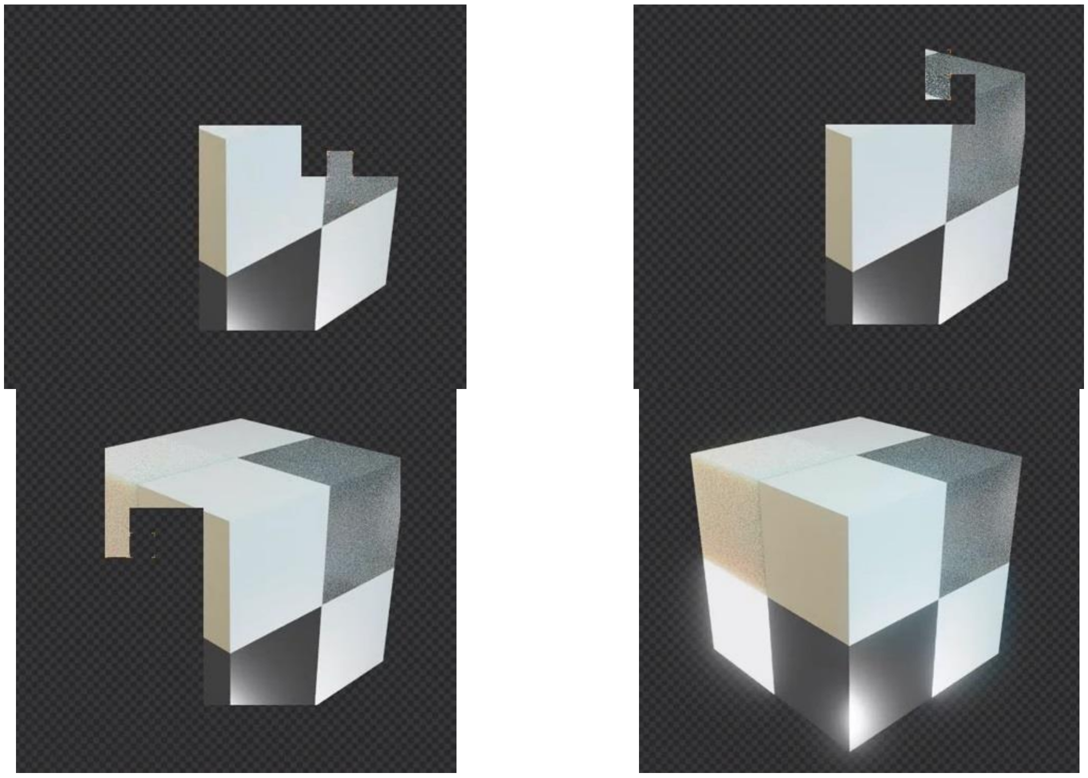
Rendering

Internal render engine with scanline rendering, indirect lighting, and ambient occlusion that can export in a wide variety of formats; a path tracer render engine called Cycles, which can take advantage of the GPU for rendering. Cycles supports the Open Shading Language since Blender 2.65. Cycles Hybrid Rendering is possible in Version 2.92 with Optix. Tiles are calculated with GPU in combination with CPU.

EEVEE is a new physically based real-time renderer. It works both as a renderer for final frames, and as the engine driving Blender's real-time viewport for creating assets.

Texture and Shading:

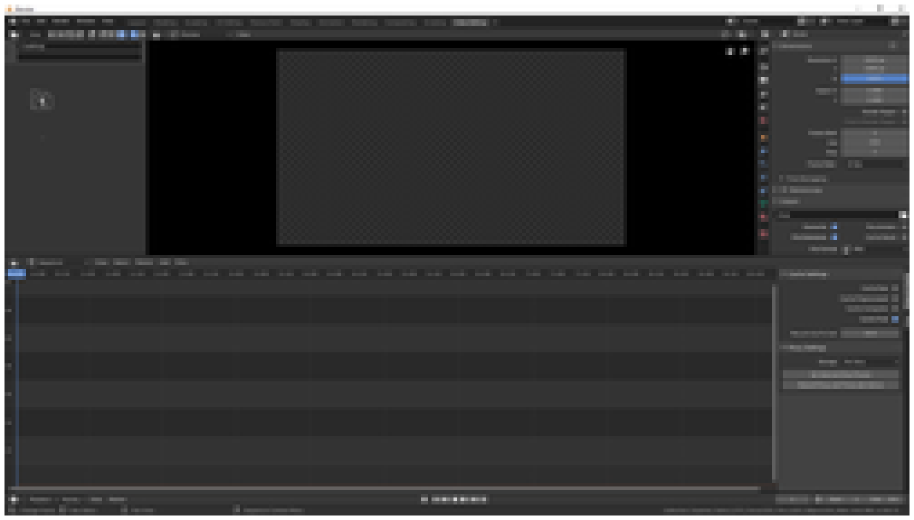
Blender allows procedural and node-based textures, as well as texture painting, projective painting, vertex painting, weight painting and dynamic painting.



Post production

Blender has a node-based compositor within the rendering pipeline accelerated with OpenCL.

Blender also includes a non-linear video editor called the Video Sequence Editor (VSE), with support for effects like Gaussian blur, color grading, fade and wipe transitions and other video transformations. However, there is no built-in multi-core support for rendering video with the VSE.



Plugins/add-ons & scripts

Blender supports Python scripting for the creation of custom tools, prototyping, game logic, importing/exporting from other formats and task automation. This allows for integration with several external render engines through plugins/add-ons.

Leap Motion

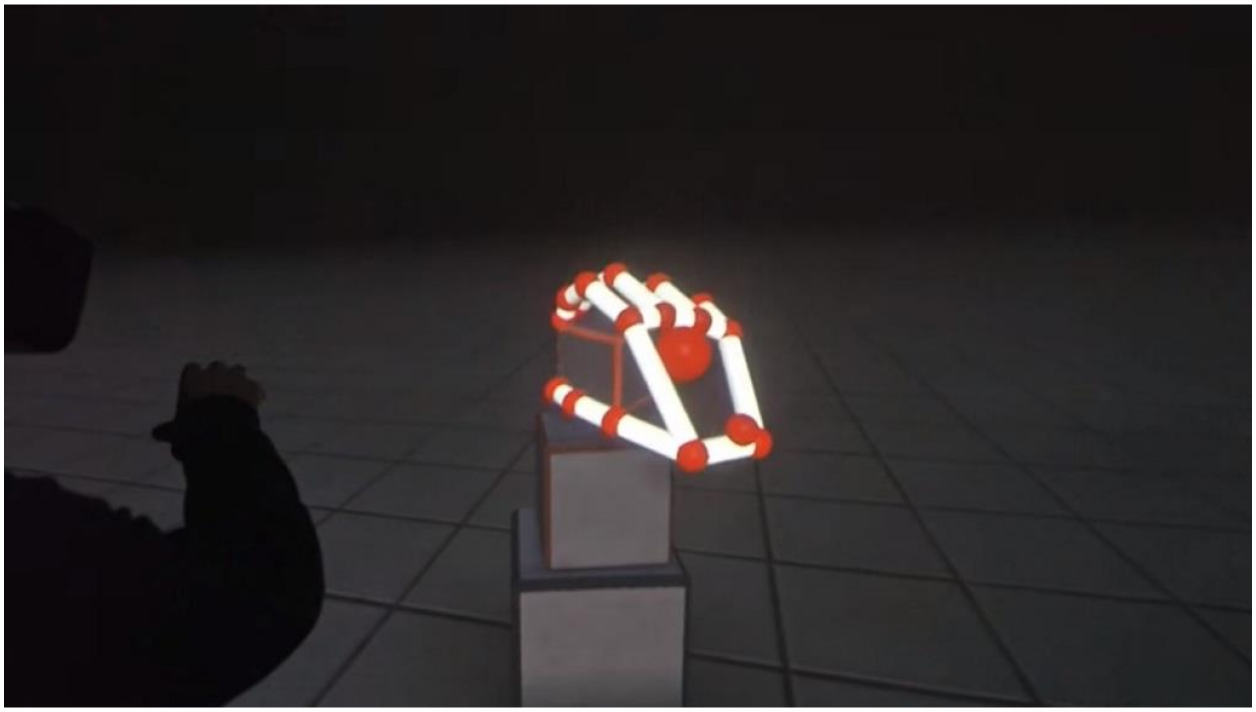
The **Leap Motion Controller** is an optical hand tracking module that captures the movement of users' hands and fingers so they can interact naturally with digital content.

Small, fast, and accurate, it can be used for productivity applications with Windows computers, integrated into enterprise-grade hardware solutions or displays, or attached to virtual/augmented reality headsets for AR/VR/XR prototyping, research, and development. The controller is capable of tracking hands within a 3D interactive zone that extends up to 60 cm (24") or more, extending from the device in a 120 x 150° field of view.

Leap Motion's software is able to discern 27 distinct hand elements, including bones and joints, and track them even when they are obscured by other parts of the hand.

Leap Motion for Virtual reality

Reach into new virtual worlds and get hands-on with VR. With a Leap Motion Controller mounted to the front of your VR headset and Leap Motion hand tracking, you can touch and interact with objects in virtual reality as you would in the real world – just by using your hands.

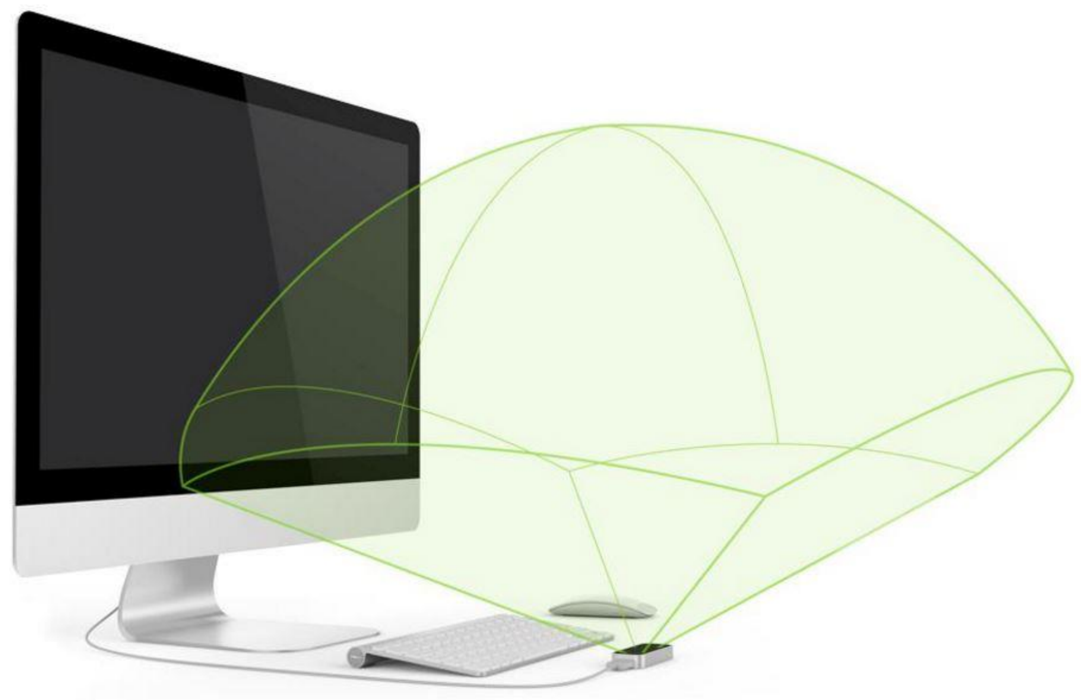


Leap Motion on your desktop

If you’re not ready to dive into VR, use the Leap Motion Controller on your desk, alongside your mouse and keyboard, for an added level of 3D interactivity. With apps from the Leap Motion Gallery, you can create music, learn about earthquakes, play games, and more.

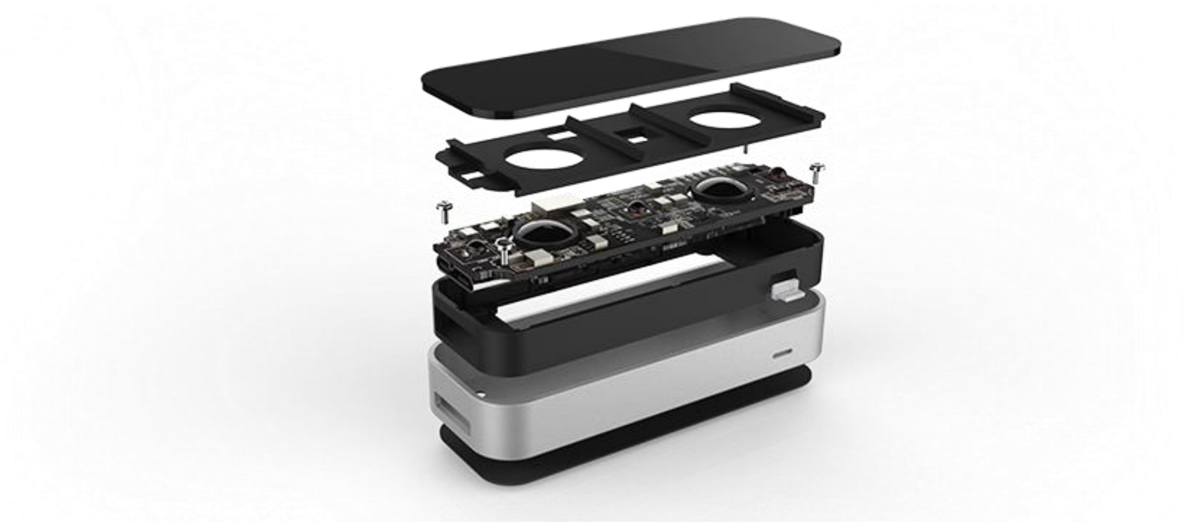
The Leap Motion ControllerTM is certified compliant to safety and electrical regulatory standards. Its robustness and external certification enable commercial projects, including sterile environments.

Experimental LeapUVC interface opens up access to low-level controls such as LED brightness, gamma, exposure, gain, resolution, and more. Use the Leap Motion Controller to track physical objects, capture high-speed infrared footage, or see the world in new ways.



How does it work?

The Leap Motion Controller uses an infrared scanner and sensor to map and track the human hand. This information is used to create, in real time, a digital version of the hand that can manipulate digital objects.



Up until this point nearly all interactions with computer programs required an intermediary step (Mouse, keyboard, etc.) between the human hand and the digital environment. The Leap Motion Controller is a big step towards bridging this gap and allowing humans to manipulate computer programs in a similar manner that they manipulate real world objects.

With the ability for developers to design their own software for the Leap Motion Controller its creative potential is incredible. The creators of Leap Motion have put the tools into the hands of others to experiment with and modify. This is already leading to the development of new, unique, software and uses for this technology.

Because Leap Motion Controllers allow users to manipulate 3D objects in an instinctual way they can be used to familiarize students with complex structures. Currently anatomy students with Leap can use software like Cyber Science 3D to dissect a body and chemistry students can examine molecules from the RCSB protein bank using the Molecules program. Both are just a few examples of the educational benefits of gesture based computing.

Scope

The scope of this project is to build a gaming platform that utilizes Unity graphics. It will support leap motion controller as an input peripheral.

Game will feature multiple levels requiring players to learn different concepts of Physics such as Range, Velocity, Force and Acceleration etc. hence providing players a platform where learning and teaching can be made fun and easy.

The game will be built on Unity3D using Leap motion as controller which will give players a realistic experience interacting with the 3D world.

The system can support running single game at a time. The system won't support saving games. If there is time, other functionalities like replay, more options, and more levels will be included.

Game Engine	Unity3D
Platforms	Windows, MAC
Language Support	C#, UnityScript
Physics Engine	PhysX
Forward Compatibility	Partial
Backward Compatibility	Yes

Comparison

	Serious Games By Designing Digital Inc.	**Our Game**
Audience	Employees	Students
Purpose	Training & skill development	Teaching & learning
Budget	Expensive	Cheap
Market	Corporate	Schools
Research	Domain related research	Curriculum available on internet

Functional Requirements

Basics:

FR-B-01	The system shall detect a collision between a dynamic object and a dynamic object.
FR-B-02	The system shall detect a collision between a dynamic object and a static object.
FR-B-03	The system shall detect a collision between a dynamic object and a ground object.
FR-B-04	The system shall prevent a dynamic object to go through a dynamic object.
FR-B-05	The system shall prevent a dynamic object to go through a static object.
FR-B-06	The system shall set resistance for each ground object.
FR-B-07	The system shall only display the environment which is within a certain radius of the player.

Development:

FR-D-01	The game must be implemented with C# script
FR-D-02	The game must be developed in Unity3d
FR-D-03	Display, Control and Audio
FR-D-04	The game must be controlled with Leap Motion controller
FR-D-05	Game must help user to build strategy
FR-D-06	Game must be First person

Performance:

FR-P-01	Client should be able to render the scene at 30 fps
FR-P-02	Game should load within 1 minute
FR-P-03	Visual response to user input should be within 100 milliseconds
FR-P-04	The game must be compatible with Leap Motion controller

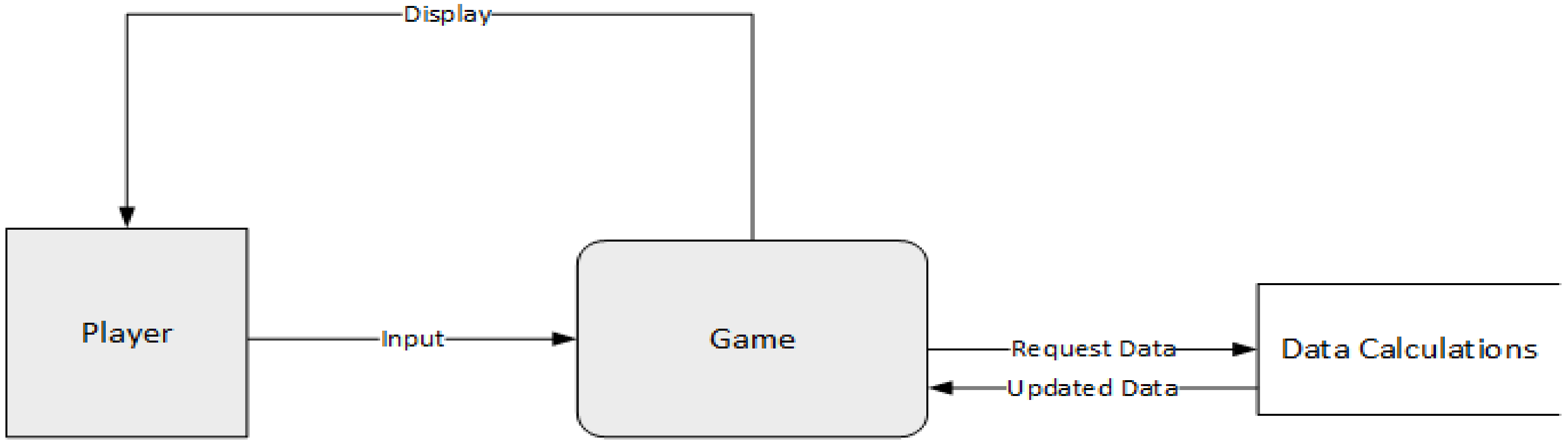
System:

FR-S-01	The system shall allow a user to be a player
FR-S-02	The system shall allow a user to be in only one game or module at the time
FR-S-03	The system shall allow a user to reset the game environment
FR-S-04	The system shall allow a user to start the game
FR-S-05	The system shall allow a user to end the game
FR-S-06	The system should show hints to play the game
FR-S-07	The system should calculate score

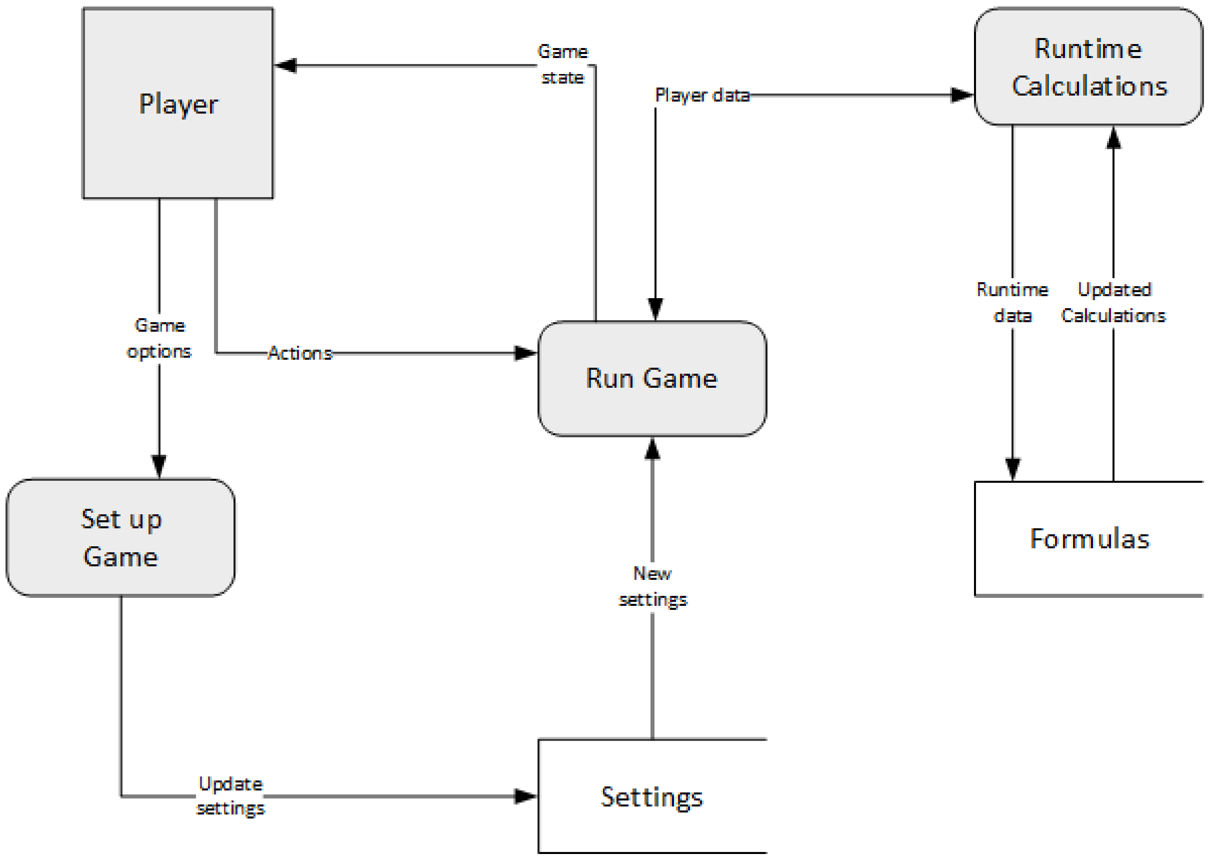
Non Functional Requirements

User Interface	
NFR-UI-01	User interface should be clean and compatible
NFR-UI-02	UI must be controlled through Leap Motion
Software Interfaces	
NFR-SI-01	UNITY will act as a bridge between Game and Leap Motion Controller
NFR-SI-02	Game must help user to build strategy
NFR-SI-03	Response Time between click & reaction must be less than 0.5 seconds
NFR-SI-04	Minimum frame rate must be 15 fps
NFR-SI-05	Game must be able to run with 1024 MB of RAM
NFR-SI-06	Game must run on Windows 8.1 or greater
NFR-SI-07	Expandability: Adding new modules & features and changing old ones should be easy

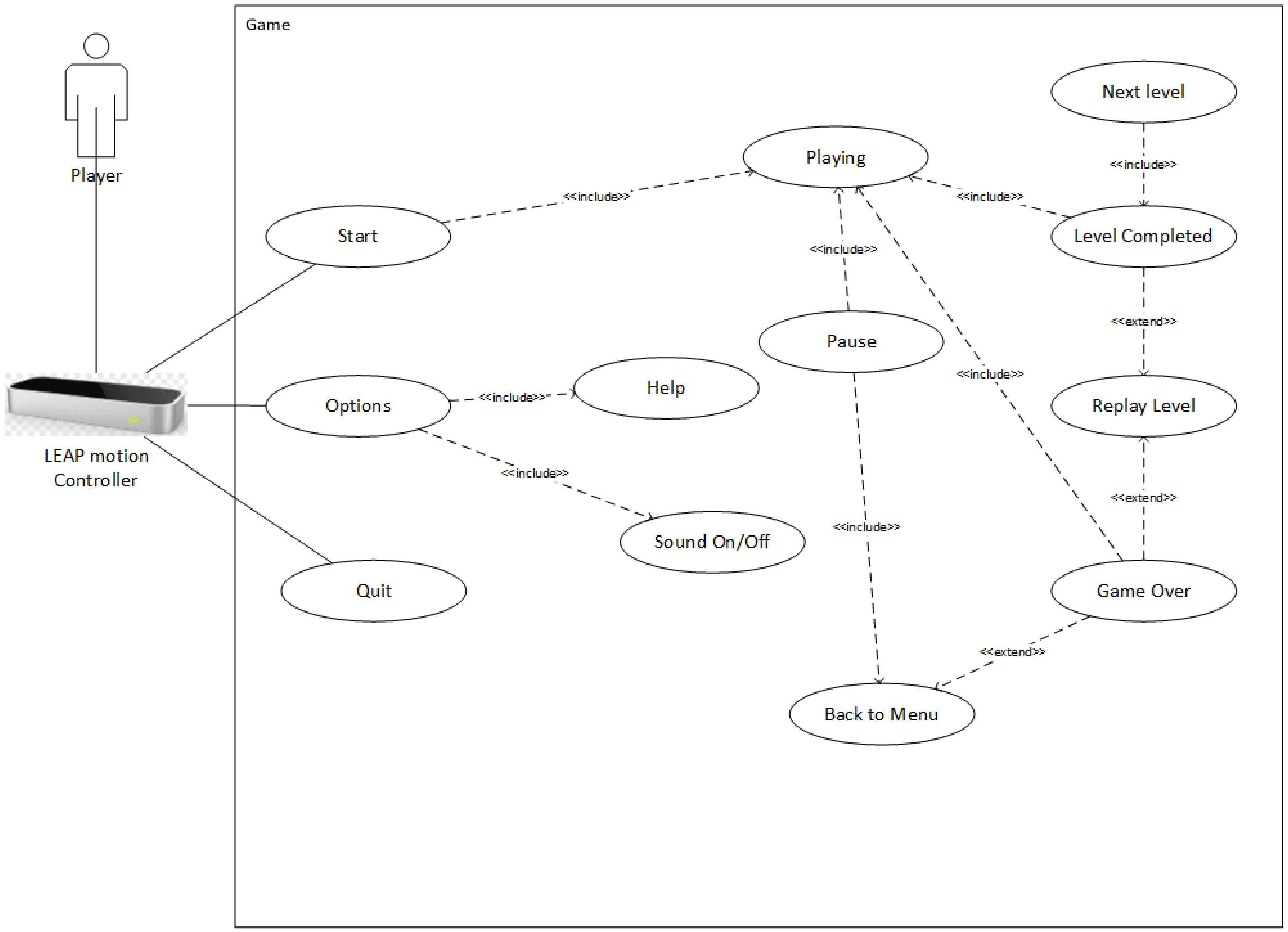
Data Flow Diagram
Level 0



Data Flow Diagram
Level 1

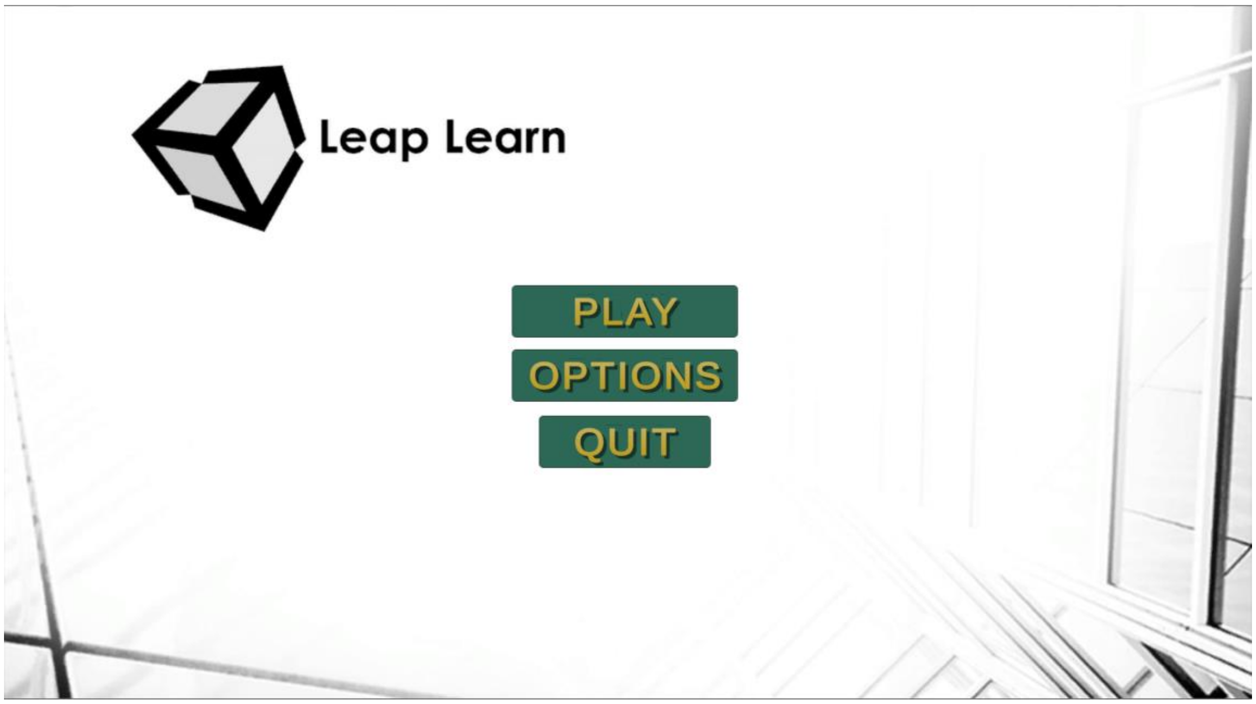


Use Case Diagram



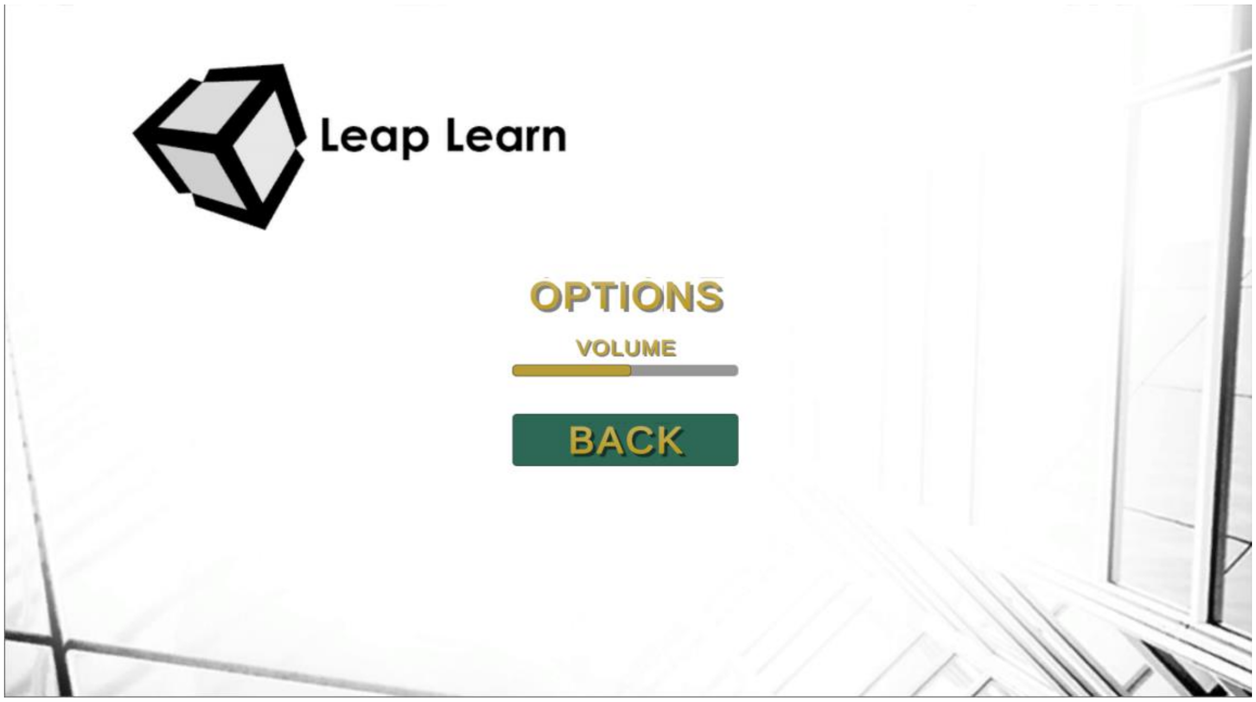
Game Design & Development

Main Menu:



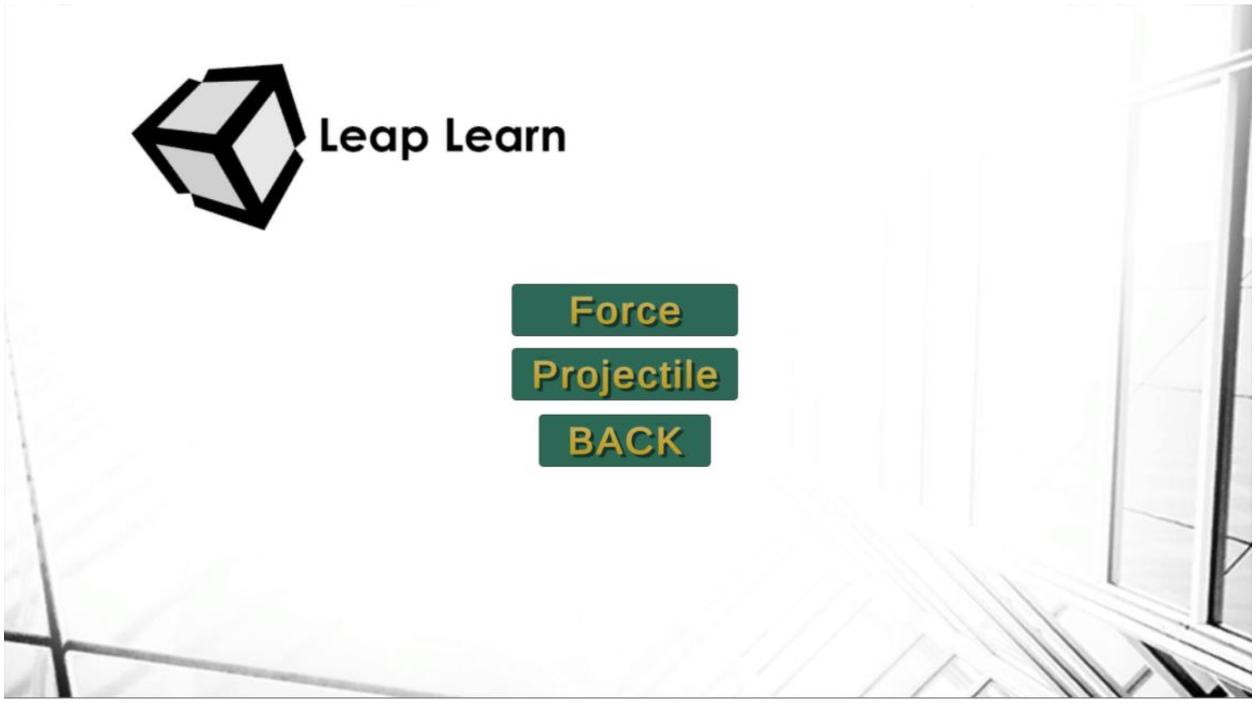
- “Play” button opens another menu where player can select game scene.
- “Options” button opens volume control.
- “Quit” button quits or exits the application.

Options:



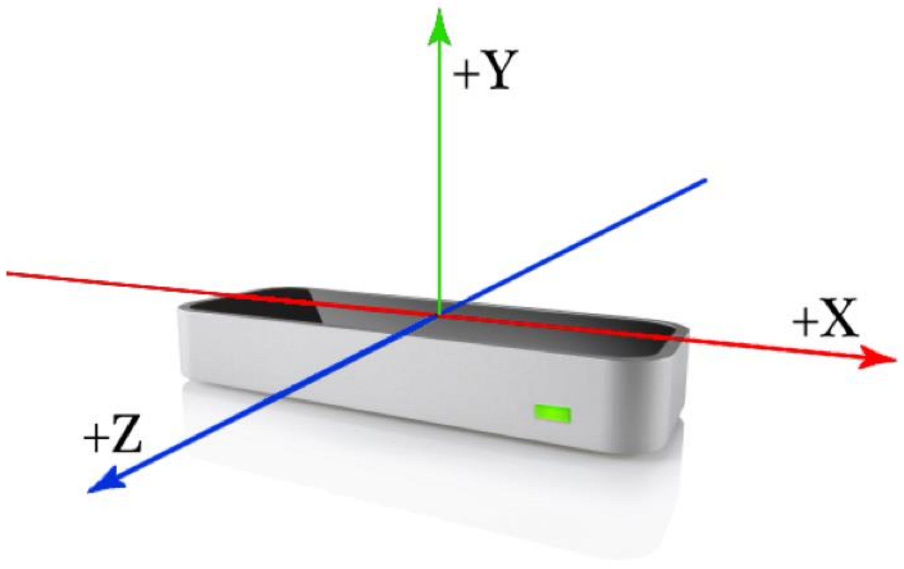
- “Slider” controls the max or low functionality.
- “Back” button saves the adjusted volume & takes player back to Main Menu.

Play Menu:



- “Force” button opens game scene “Force”.
- “Projectile” button opens game scene “Projectile”.
- “Back” button takes player to Main Menu.

Controls:



Z-axis – Player hand behind the controller will move character backward

Z-axis – Player hand in front of the controller will move character forward

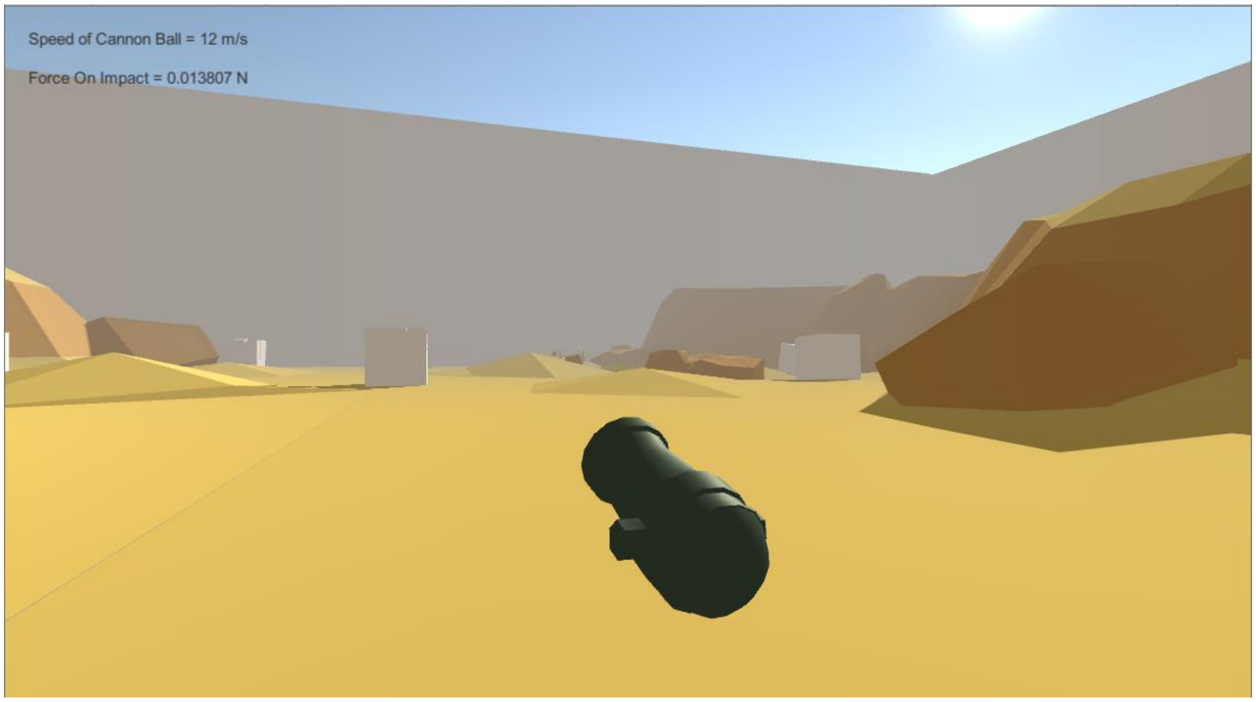
X-axis – Player hand right of the controller will move character towards right

X-axis – Player hand left of the controller will move character towards left

Palm Direction – Camera will rotate according to palm movement

Note: If there are two hands in scene, the hand near to player will control movement

Force:



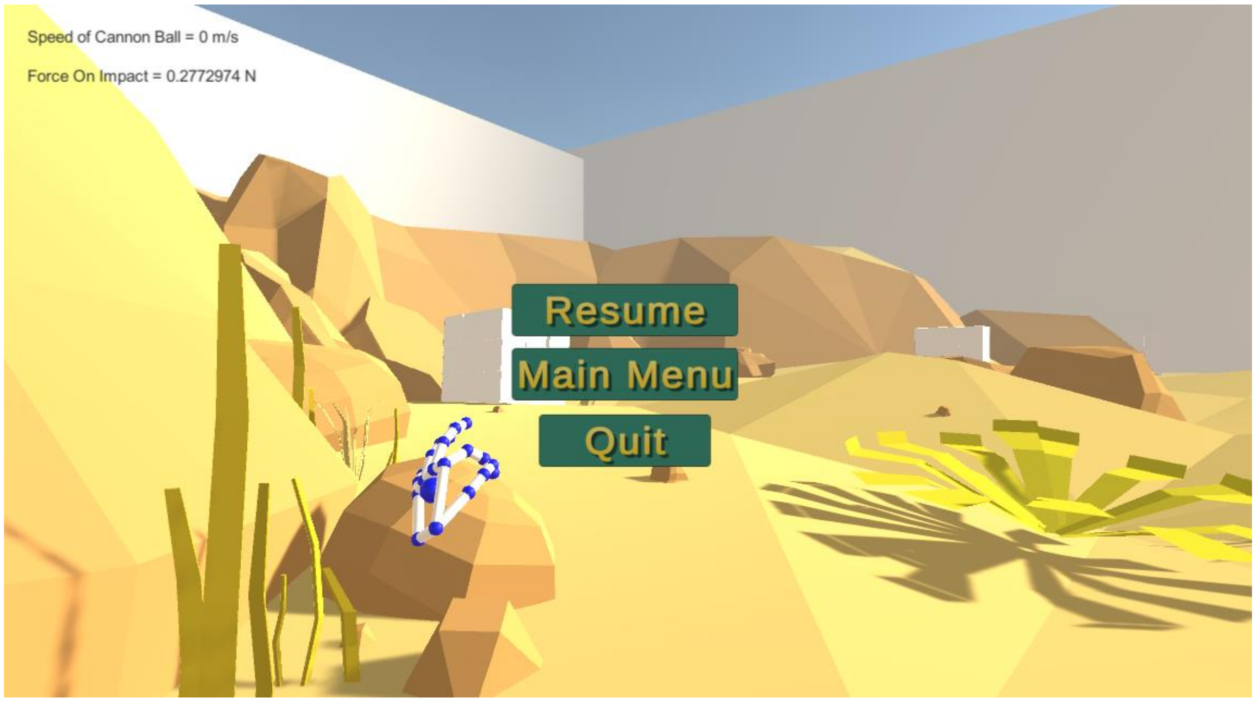
UI:

- **Speed of Cannon ball** shows the speed of cannon ball or shot speed.
- **Force on Impact** shows the Force exerted on boxes when cannon ball hits them.

Scene Controls:

- Cannon shoots when it detects a pinch in right hand.

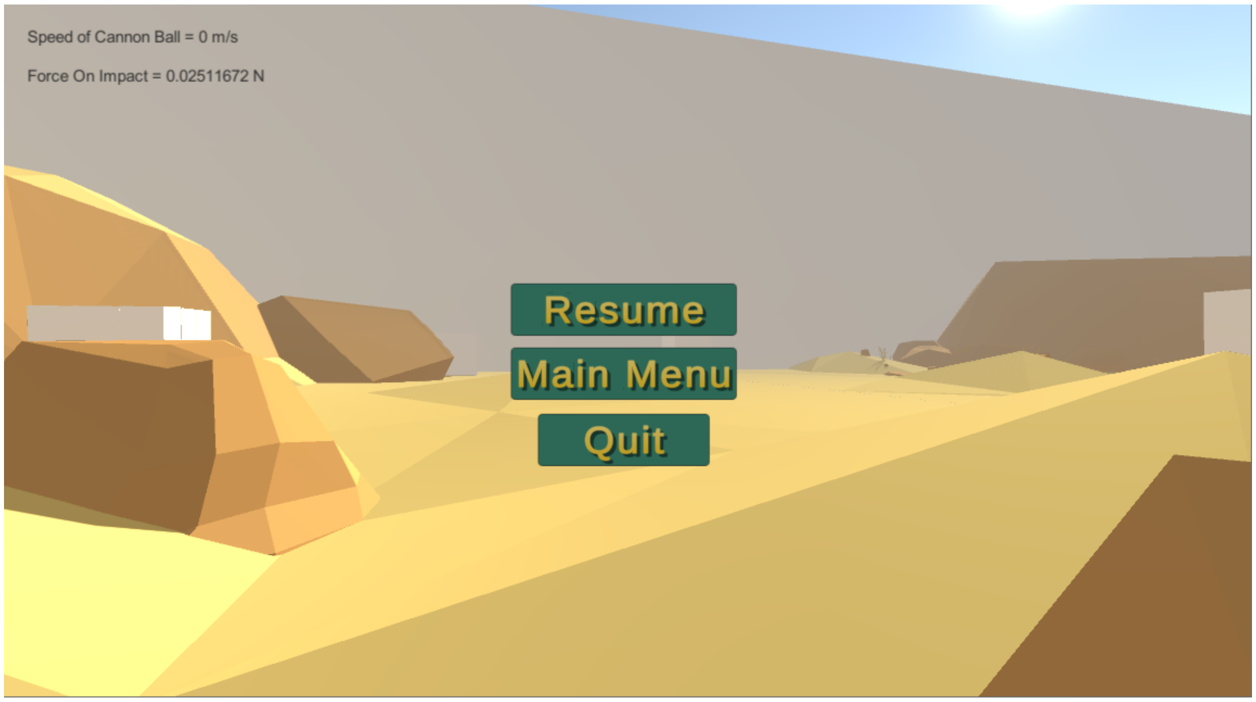
Force:



Scene Controls:

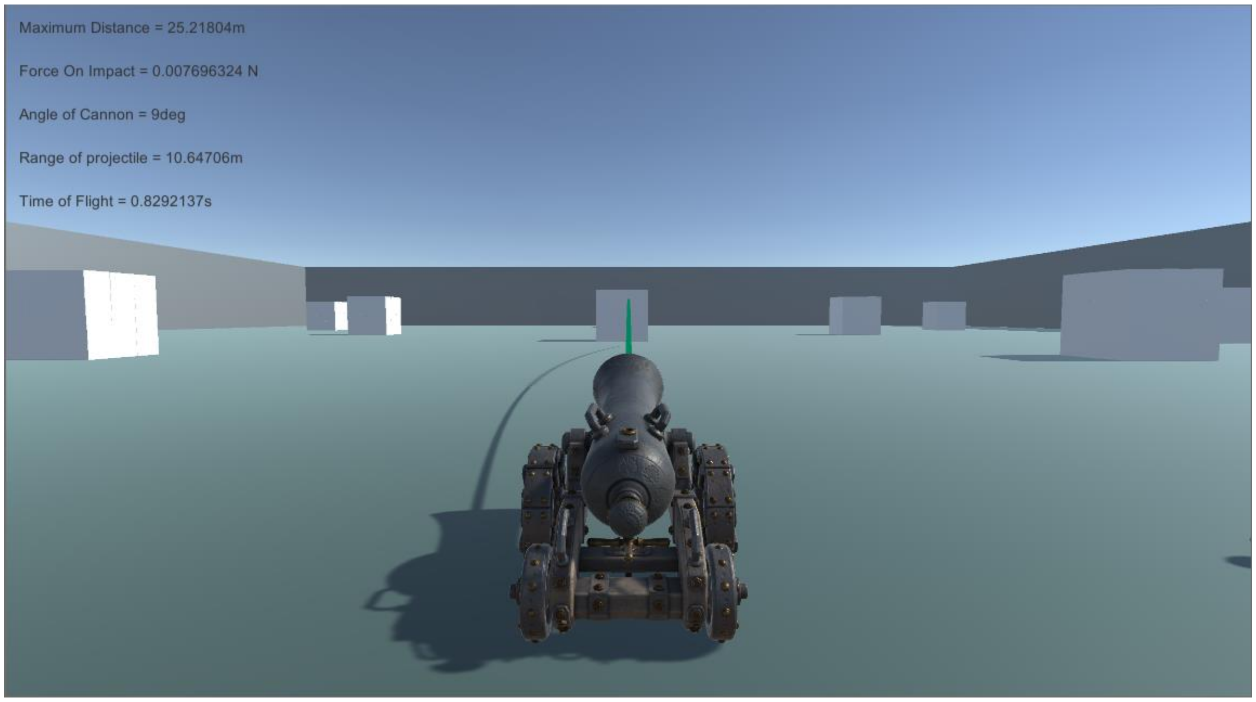
- Game is paused & pause menu pops-up when a pinch is detected in left hand.

Pause Menu:



- "Resume" button opens resumes the game scene.
- "Main Menu" button directs player to Main Menu.
- "Quit" button quits or exits the application.

Projectile:


Maximum Distance = 25.21804m

Force On Impact = 0.007696324 N

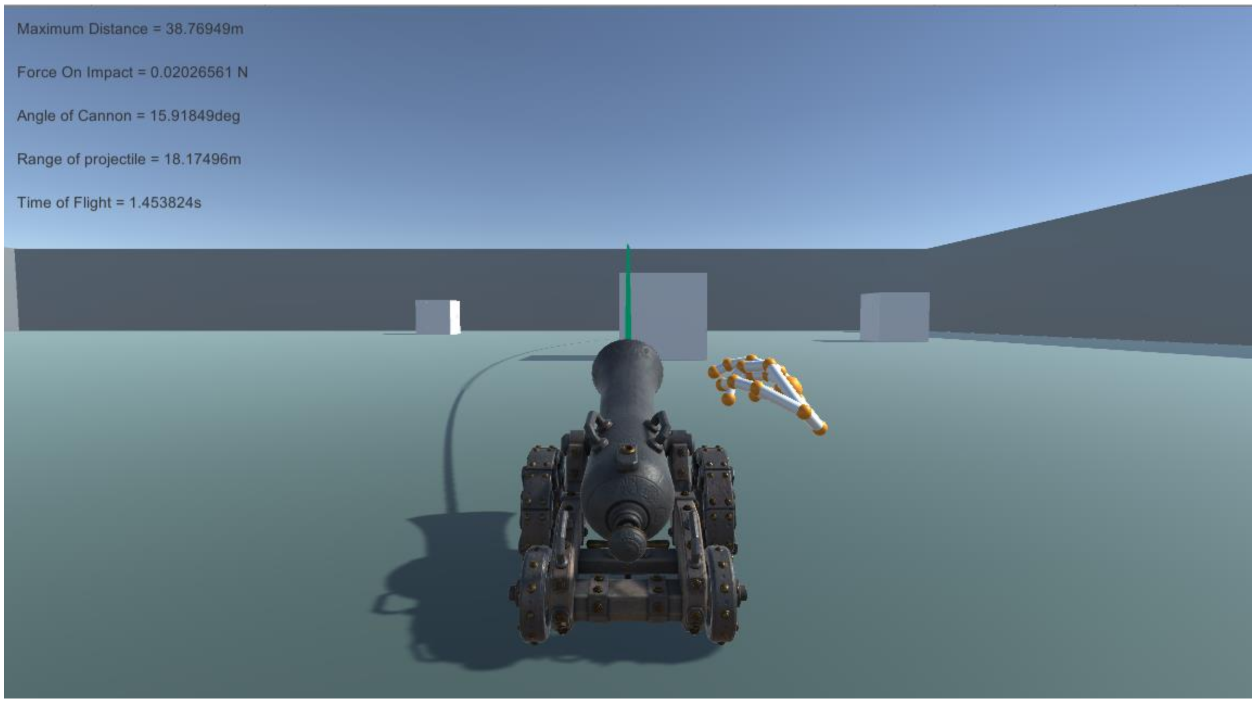
Angle of Cannon = 9deg

Range of projectile = 10.64706m

Time of Flight = 0.8292137s

- **Maximum Distance** shows the maximum distance which the cannon ball would cover.
- **Force on Impact** shows the Force exerted on boxes when cannon ball hits them.
- **Angle of Cannon** shows the cannon’s barrel angle with respect to horizon.
- **Range of Projectile** shows the curve or parabolic distance.
- **Time of Flight** shows the time which the cannon ball will take to reach its target.

Projectile:



Maximum Distance = 38.76949m

Force On Impact = 0.02026561 N

Angle of Cannon = 15.91849deg

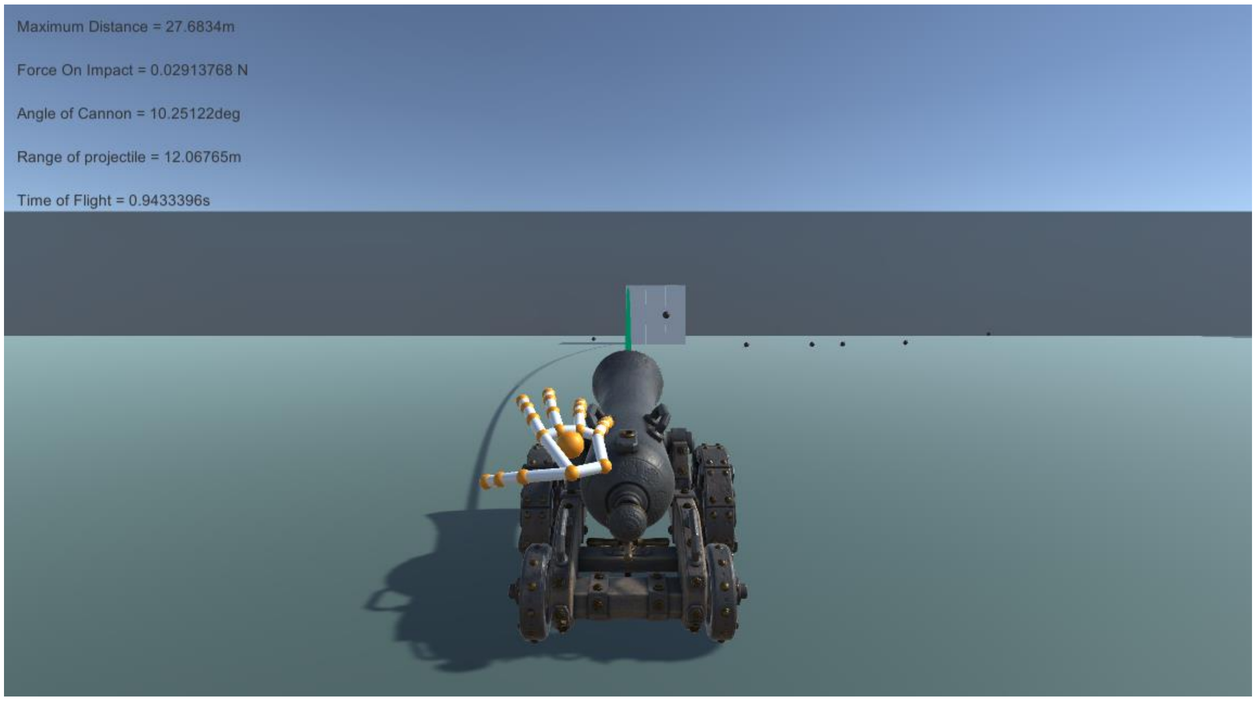
Range of projectile = 18.17496m

Time of Flight = 1.453824s

Scene Controls:

- Cannon rotates up when it detects a pinch in right hand.

Projectile:

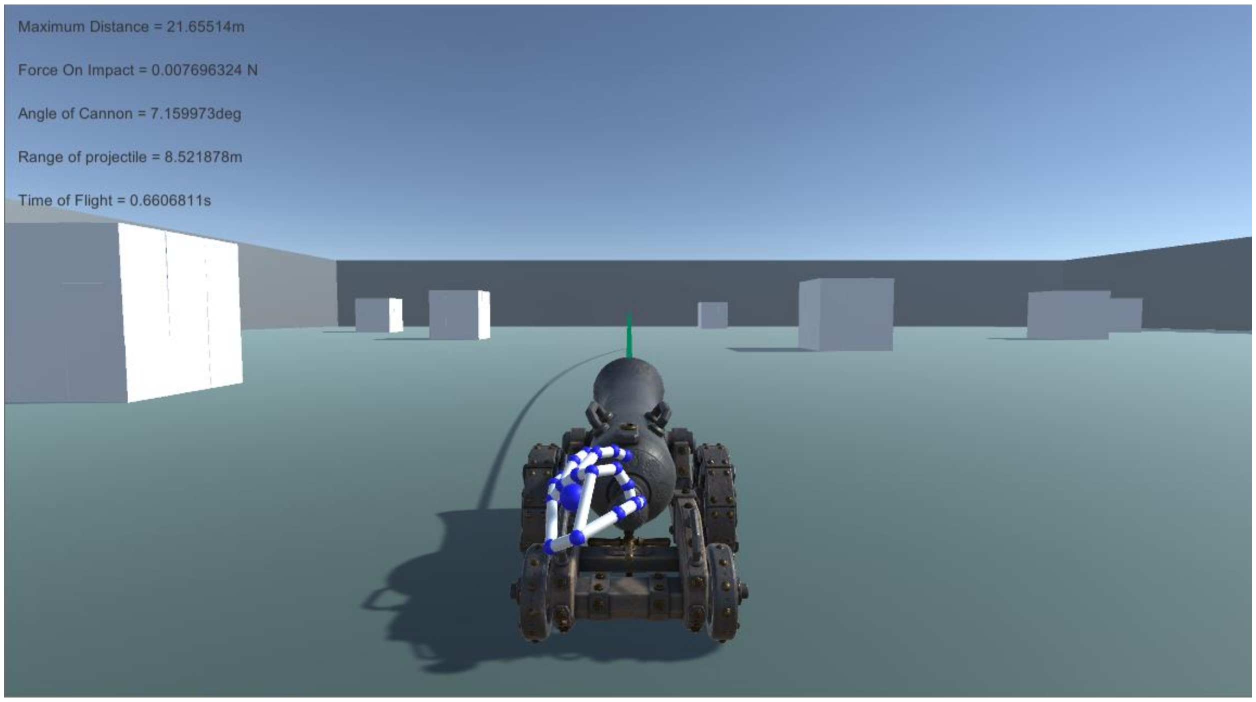


```
Maximum Distance = 27.6834m

Force On Impact = 0.02913768 N

Angle of Cannon = 10.25122deg

Range of projectile = 12.06765m

Time of Flight = 0.9433396s
```

Scene Controls:

- Cannon shoots when it detects an open hand of right hand.

Projectile:

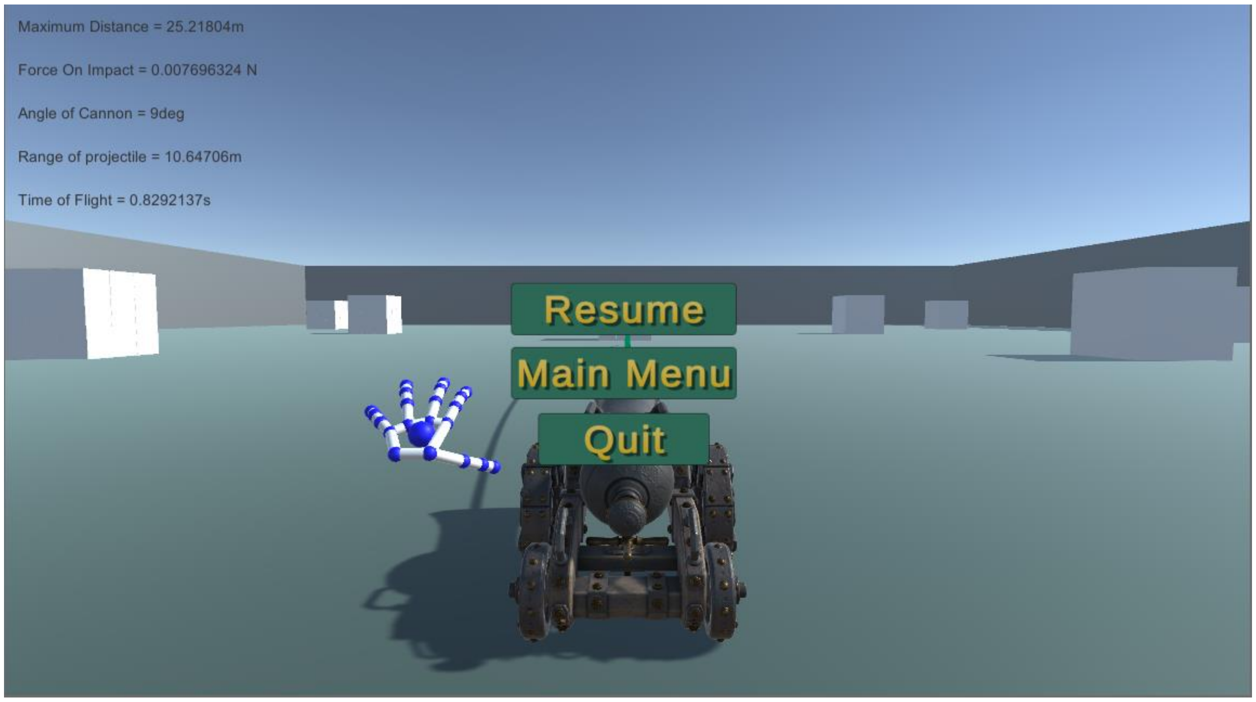


```
Maximum Distance = 21.65514m

Force On Impact = 0.007696324 N

Angle of Cannon = 7.159973deg

Range of projectile = 8.521878m

Time of Flight = 0.6606811s
```

Scene Controls:

- Cannon rotates down when it detects a pinch in left hand.

Projectile:



Maximum Distance = 25.21804m

Force On Impact = 0.007696324 N

Angle of Cannon = 9deg

Range of projectile = 10.64706m

Time of Flight = 0.8292137s

Scene Controls:

- Game is paused & pause menu pops-up when open hand is detected in left hand.

Formulas

Projectile Motion

Projectile motion is a form of motion where an object moves in a bilaterally symmetrical, parabolic path. The path that the object follows is called its trajectory. Projectile motion only occurs when there is one force applied at the beginning on the trajectory, after which the only interference is from gravity. In a previous atom we discussed what the various components of an object in projectile motion are. In this atom we will discuss the basic equations that go along with them in the special case in which the projectile initial positions are null (i.e. x0=0 and y0=0).

Initial Velocity

The initial velocity can be expressed as x components and y components:

$$U_x = u.cos\theta$$

$$U_y = u.sin\theta$$

In this equation, u stands for initial velocity magnitude and θ refers to projectile angle.

Time of Flight

The time of flight of a projectile motion is the time from when the object is projected to the time it reaches the surface. As we discussed previously, TT depends on the initial velocity magnitude and the angle of the projectile:

$$\mathrm{T} = \frac{2.u_y}{g}$$

$$\mathrm{T} = \frac{2.\mathrm{u}.\sin\theta}{g}$$

Acceleration

In projectile motion, there is no acceleration in the horizontal direction. The acceleration, a, in the vertical direction is just due to gravity, also known as free fall:

$$a_x = 0$$

$$a_y = -g$$

Velocity

The horizontal velocity remains constant, but the vertical velocity varies linearly, because the acceleration is constant. At any time, t, the velocity is:

$$U_x = u.\,cos\theta$$

$$U_y = u.\,sin\theta - g\,.\,t$$

You can also use the Pythagorean Theorem to find velocity:

$$\sqrt{x^2 + y^2} = \Delta r^2$$

Displacement

At time, t, the displacement components are:

$$x = u.\,t.\,cos\theta$$

$$y = u.\,t.\,sin\theta - \frac{1}{2}.\,g.\,t^2$$

The equation for the magnitude of the displacement is

$$\sqrt{x^2 + y^2} = \Delta r^2$$

Parabolic Trajectory

We can use the displacement equations in the x and y direction to obtain an equation for the parabolic form of a projectile motion:

$$\mathrm{y} = \tan\theta\,.\,\mathrm{x} - \frac{g}{2\,.\,\mathrm{u}^2\,.\,cos^2\theta}\,.\,x^2$$

Maximum Height

The maximum height is reached when$v_y = 0$. Using this we can rearrange the velocity equation to find the time it will take for the object to reach maximum height

$$t_h = \frac{\mathrm{u.\,sin\theta}}{g}$$

Where t_h stands for the time it takes to reach maximum height. From the displacement equation we can find the maximum height

$$\mathrm{h} = \frac{u^2.sin^2\theta}{2.g}$$

Range

The range of the motion is fixed by the condition y=0y=0. Using this we can rearrange the parabolic motion equation to find the range of the motion:

$$\mathrm{R} = \frac{u^2.\sin 2\theta}{g}$$

Force

The formula for force states that force is equal to mass multiplied by acceleration.

$$F = m.a$$

Narrations

Use Case Name:	Start Game
Actor:	Player
Priority:	Essential
Trigger:	User clicks on play button on screen
Precondition:	None
Basic Path:	User clicks on play button
Alternate Path:	• None
Post condition:	The game is launched
Exception Path:	

Use Case Name:	View
Actor:	Player
Priority:	Essential
Trigger:	User clicks on play button
Precondition:	Use Case Start Game has been executed
Basic Path:	1. Player can see the dynamic objects 2. Player can see the static objects 3. Player can see the ground objects
Alternate Path:	• None
Post condition:	Environment is visible
Exception Path:	

Use Case Name:	Input
Actor:	Device
Priority:	Essential
Trigger:	Leap Motion Controller is connected
Precondition:	Game is built to read input from controller
Basic Path:	Player interacts with environment and objects through controller
Alternate Path:	• None
Post condition:	Data from controller will be used for score
Exception Path:	

Use Case Name:	Interact
Actor:	Player
Priority:	Essential
Trigger:	User performs action
Precondition:	1. Game's environment is visible 2. Leap Motion controller is connected
Basic Path:	Player can interact with environment and objects
Alternate Path:	• None
Post condition:	Player can touch objects in game
Exception Path:	

Use Case Name:	Options
Actor:	Player
Priority:	Essential
Trigger:	Player presses options button
Precondition:	Options menu is visible
Basic Path:	Player can choose between two options: 1. Help 2. Sound On/Off
Alternate Path:	• None
Post condition:	Options menu is visible
Exception Path:	

Use Case Name:	Options - Help
Actor:	Player
Priority:	Feature
Trigger:	Player presses help button
Precondition:	Options menu is visible
Basic Path:	Player will be guided throughout the game about how game works & what targets are to be achieved
Alternate Path:	• None
Post condition:	Player can see hints
Exception Path:	

Use Case Name:	Options - Sound
Actor:	Player
Priority:	Feature
Trigger:	Player presses sound on/off button
Precondition:	Options menu is visible
Basic Path:	Player can turn on/off sound
Alternate Path:	• None
Post condition:	Player can hear sound Player can stop sound
Exception Path:	

Use Case Name:	Pause
Actor:	Player
Priority:	Essential
Trigger:	Player presses pause button
Precondition:	Player must be in the game
Basic Path:	Player can press pause button on screen
Alternate Path:	• None
Post condition:	Game is paused
Exception Path:	

Use Case Name:	Score
Actor:	Player
Priority:	Essential
Trigger:	Ball hits the target
Precondition:	Player must throw the ball
Basic Path:	Player throws the ball & it hits the target
Alternate Path:	• None
Post condition:	Player sees their score
Exception Path:	

Use Case Name:	Score - formula
Actor:	Player
Priority:	Essential
Trigger:	Ball hits the target
Precondition:	Use Case - Score
Basic Path:	Physics formulas are implemented on score
Alternate Path:	• None
Post condition:	Player sees their score
Exception Path:	

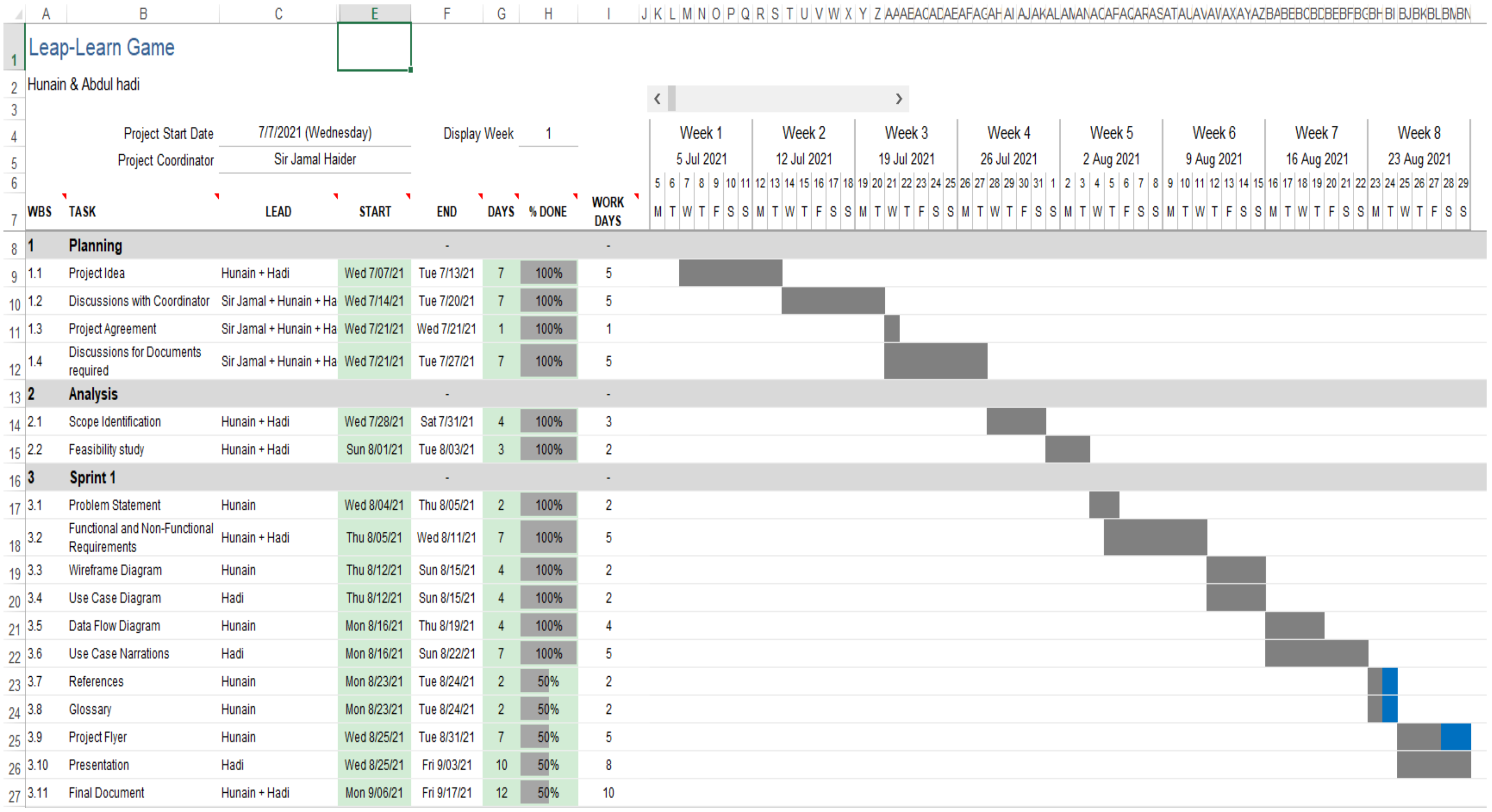
Use Case Name:	Level completed
Actor:	Player
Priority:	Essential
Trigger:	Level end
Precondition:	Player must pass the level
Basic Path:	Player passes the level by achieving goal
Alternate Path:	• None
Post condition:	Next level Replay level
Exception Path:	

Use Case Name:	Next level
Actor:	Player
Priority:	Essential
Trigger:	Level end
Precondition:	Level is successfully cleared
Basic Path:	Next level is loaded
Alternate Path:	• None
Post condition:	Player sees a new level
Exception Path:	

Use Case Name:	Game over
Actor:	Player
Priority:	Essential
Trigger:	Player loses the level
Precondition:	Player must be in the game
Basic Path:	Player is not able to achieve the goal
Alternate Path:	• None
Post condition:	Replay level Back to menu
Exception Path:	

Use Case Name:	Exit
Actor:	Player
Priority:	Essential
Trigger:	Player stops playing game & presses exit button
Precondition:	Player must be in the game
Basic Path:	Player can press exit button on top right corner of screen
Alternate Path:	• None
Post condition:	Game closed
Exception Path:	

Gantt chart



Leap-Learn Game

Hunain & Abdul hadi

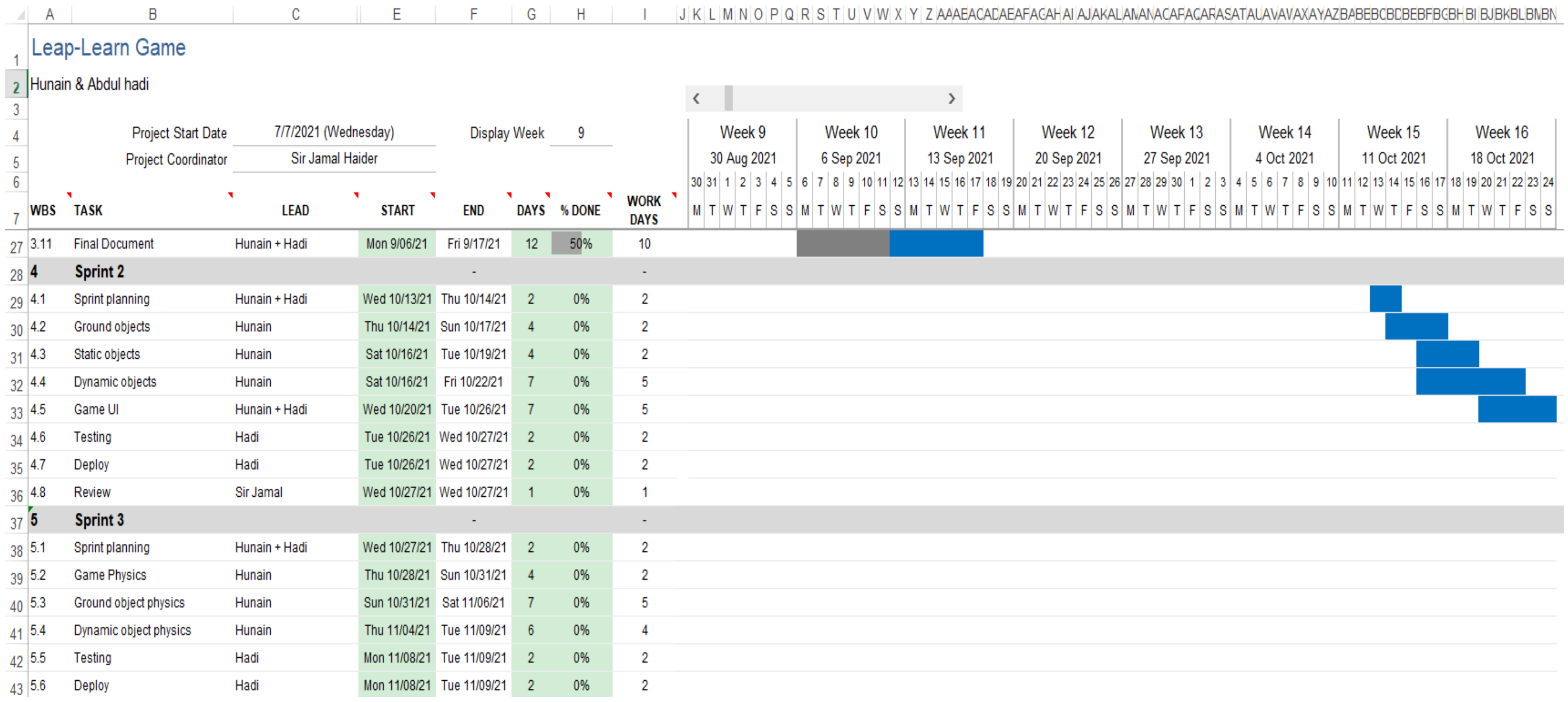
Project Start Date: 7/7/2021 (Wednesday)

Display Week: 1

Project Coordinator: Sir Jamal Haider

Week 1 5 Jul 2021 | Week 2 12 Jul 2021 | Week 3 19 Jul 2021 | Week 4 26 Jul 2021 | Week 5 2 Aug 2021 | Week 6 9 Aug 2021 | Week 7 16 Aug 2021 | Week 8 23 Aug 2021

WBS	TASK	LEAD	START	END	DAYS	% DONE	WORK DAYS
1	**Planning**			-			-
1.1	Project Idea	Hunain + Hadi	Wed 7/07/21	Tue 7/13/21	7	100%	5
1.2	Discussions with Coordinator	Sir Jamal + Hunain + Ha	Wed 7/14/21	Tue 7/20/21	7	100%	5
1.3	Project Agreement	Sir Jamal + Hunain + Ha	Wed 7/21/21	Wed 7/21/21	1	100%	1
1.4	Discussions for Documents required	Sir Jamal + Hunain + Ha	Wed 7/21/21	Tue 7/27/21	7	100%	5
2	**Analysis**			-			-
2.1	Scope Identification	Hunain + Hadi	Wed 7/28/21	Sat 7/31/21	4	100%	3
2.2	Feasibility study	Hunain + Hadi	Sun 8/01/21	Tue 8/03/21	3	100%	2
3	**Sprint 1**			-			-
3.1	Problem Statement	Hunain	Wed 8/04/21	Thu 8/05/21	2	100%	2
3.2	Functional and Non-Functional Requirements	Hunain + Hadi	Thu 8/05/21	Wed 8/11/21	7	100%	5
3.3	Wireframe Diagram	Hunain	Thu 8/12/21	Sun 8/15/21	4	100%	2
3.4	Use Case Diagram	Hadi	Thu 8/12/21	Sun 8/15/21	4	100%	2
3.5	Data Flow Diagram	Hunain	Mon 8/16/21	Thu 8/19/21	4	100%	4
3.6	Use Case Narrations	Hadi	Mon 8/16/21	Sun 8/22/21	7	100%	5
3.7	References	Hunain	Mon 8/23/21	Tue 8/24/21	2	50%	2
3.8	Glossary	Hunain	Mon 8/23/21	Tue 8/24/21	2	50%	2
3.9	Project Flyer	Hunain	Wed 8/25/21	Tue 8/31/21	7	50%	5
3.10	Presentation	Hadi	Wed 8/25/21	Fri 9/03/21	10	50%	8
3.11	Final Document	Hunain + Hadi	Mon 9/06/21	Fri 9/17/21	12	50%	10



Leap-Learn Game

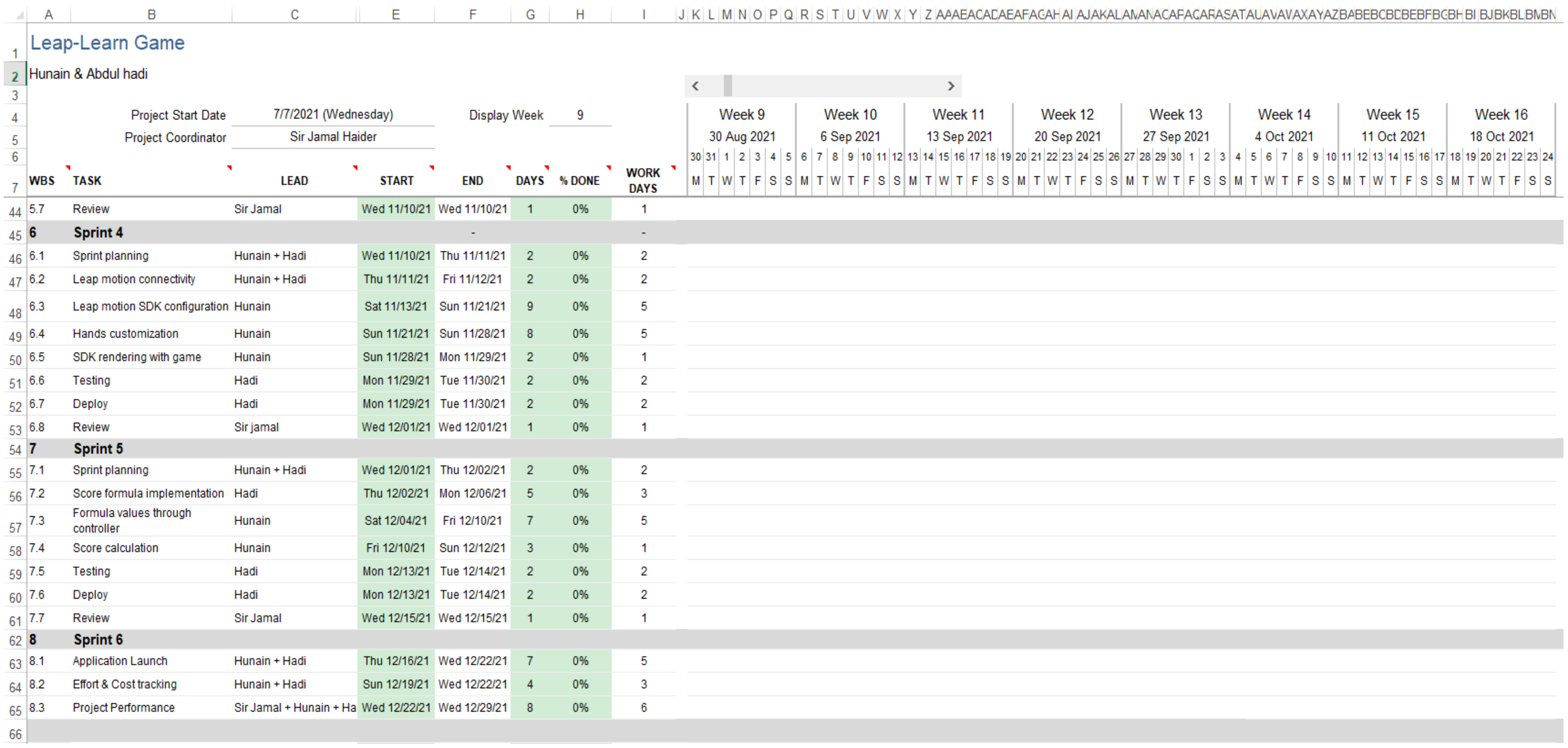
Hunain & Abdul hadi

Project Start Date: 7/7/2021 (Wednesday) Display Week: 9

Project Coordinator: Sir Jamal Haider

Week 9 30 Aug 2021 | Week 10 6 Sep 2021 | Week 11 13 Sep 2021 | Week 12 20 Sep 2021 | Week 13 27 Sep 2021 | Week 14 4 Oct 2021 | Week 15 11 Oct 2021 | Week 16 18 Oct 2021

WBS	TASK	LEAD	START	END	DAYS	% DONE	WORK DAYS
3.11	Final Document	Hunain + Hadi	Mon 9/06/21	Fri 9/17/21	12	50%	10
4	**Sprint 2**			-			-
4.1	Sprint planning	Hunain + Hadi	Wed 10/13/21	Thu 10/14/21	2	0%	2
4.2	Ground objects	Hunain	Thu 10/14/21	Sun 10/17/21	4	0%	2
4.3	Static objects	Hunain	Sat 10/16/21	Tue 10/19/21	4	0%	2
4.4	Dynamic objects	Hunain	Sat 10/16/21	Fri 10/22/21	7	0%	5
4.5	Game UI	Hunain + Hadi	Wed 10/20/21	Tue 10/26/21	7	0%	5
4.6	Testing	Hadi	Tue 10/26/21	Wed 10/27/21	2	0%	2
4.7	Deploy	Hadi	Tue 10/26/21	Wed 10/27/21	2	0%	2
4.8	Review	Sir Jamal	Wed 10/27/21	Wed 10/27/21	1	0%	1
5	**Sprint 3**			-			-
5.1	Sprint planning	Hunain + Hadi	Wed 10/27/21	Thu 10/28/21	2	0%	2
5.2	Game Physics	Hunain	Thu 10/28/21	Sun 10/31/21	4	0%	2
5.3	Ground object physics	Hunain	Sun 10/31/21	Sat 11/06/21	7	0%	5
5.4	Dynamic object physics	Hunain	Thu 11/04/21	Tue 11/09/21	6	0%	4
5.5	Testing	Hadi	Mon 11/08/21	Tue 11/09/21	2	0%	2
5.6	Deploy	Hadi	Mon 11/08/21	Tue 11/09/21	2	0%	2



Leap-Learn Game

Hunain & Abdul hadi

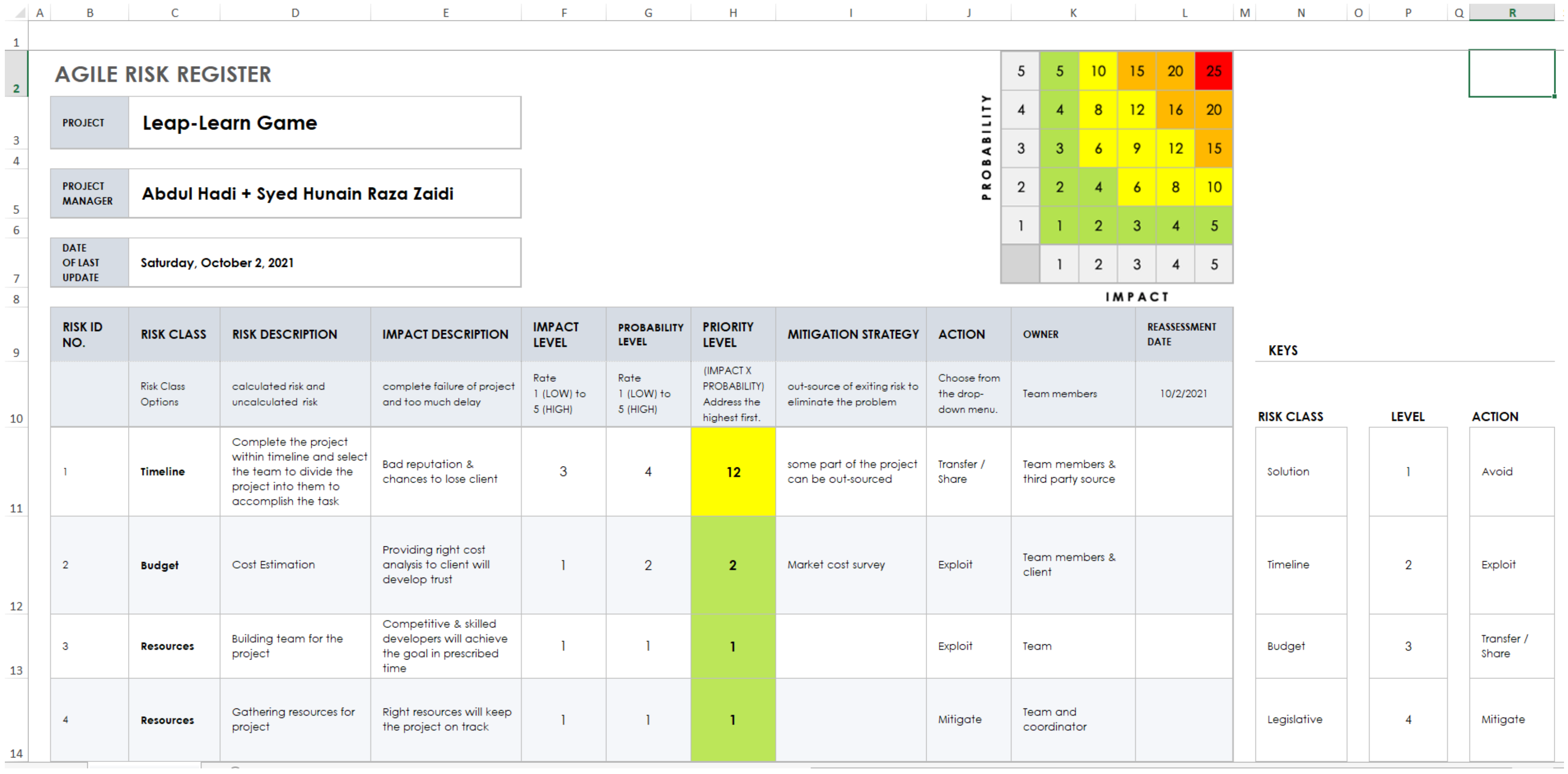
Project Start Date: 7/7/2021 (Wednesday) Display Week: 9

Project Coordinator: Sir Jamal Haider

Week 9 — 30 Aug 2021 | Week 10 — 6 Sep 2021 | Week 11 — 13 Sep 2021 | Week 12 — 20 Sep 2021 | Week 13 — 27 Sep 2021 | Week 14 — 4 Oct 2021 | Week 15 — 11 Oct 2021 | Week 16 — 18 Oct 2021

	WBS	TASK	LEAD	START	END	DAYS	% DONE	WORK DAYS
44	5.7	Review	Sir Jamal	Wed 11/10/21	Wed 11/10/21	1	0%	1
45	**6**	**Sprint 4**			-			-
46	6.1	Sprint planning	Hunain + Hadi	Wed 11/10/21	Thu 11/11/21	2	0%	2
47	6.2	Leap motion connectivity	Hunain + Hadi	Thu 11/11/21	Fri 11/12/21	2	0%	2
48	6.3	Leap motion SDK configuration	Hunain	Sat 11/13/21	Sun 11/21/21	9	0%	5
49	6.4	Hands customization	Hunain	Sun 11/21/21	Sun 11/28/21	8	0%	5
50	6.5	SDK rendering with game	Hunain	Sun 11/28/21	Mon 11/29/21	2	0%	1
51	6.6	Testing	Hadi	Mon 11/29/21	Tue 11/30/21	2	0%	2
52	6.7	Deploy	Hadi	Mon 11/29/21	Tue 11/30/21	2	0%	2
53	6.8	Review	Sir jamal	Wed 12/01/21	Wed 12/01/21	1	0%	1
54	**7**	**Sprint 5**						
55	7.1	Sprint planning	Hunain + Hadi	Wed 12/01/21	Thu 12/02/21	2	0%	2
56	7.2	Score formula implementation	Hadi	Thu 12/02/21	Mon 12/06/21	5	0%	3
57	7.3	Formula values through controller	Hunain	Sat 12/04/21	Fri 12/10/21	7	0%	5
58	7.4	Score calculation	Hunain	Fri 12/10/21	Sun 12/12/21	3	0%	1
59	7.5	Testing	Hadi	Mon 12/13/21	Tue 12/14/21	2	0%	2
60	7.6	Deploy	Hadi	Mon 12/13/21	Tue 12/14/21	2	0%	2
61	7.7	Review	Sir Jamal	Wed 12/15/21	Wed 12/15/21	1	0%	1
62	**8**	**Sprint 6**						
63	8.1	Application Launch	Hunain + Hadi	Thu 12/16/21	Wed 12/22/21	7	0%	5
64	8.2	Effort & Cost tracking	Hunain + Hadi	Sun 12/19/21	Wed 12/22/21	4	0%	3
65	8.3	Project Performance	Sir Jamal + Hunain + Ha	Wed 12/22/21	Wed 12/29/21	8	0%	6
66								

Agile Risk Register



AGILE RISK REGISTER

PROJECT	Leap-Learn Game
PROJECT MANAGER	Abdul Hadi + Syed Hunain Raza Zaidi
DATE OF LAST UPDATE	Saturday, October 2, 2021

PROBABILITY

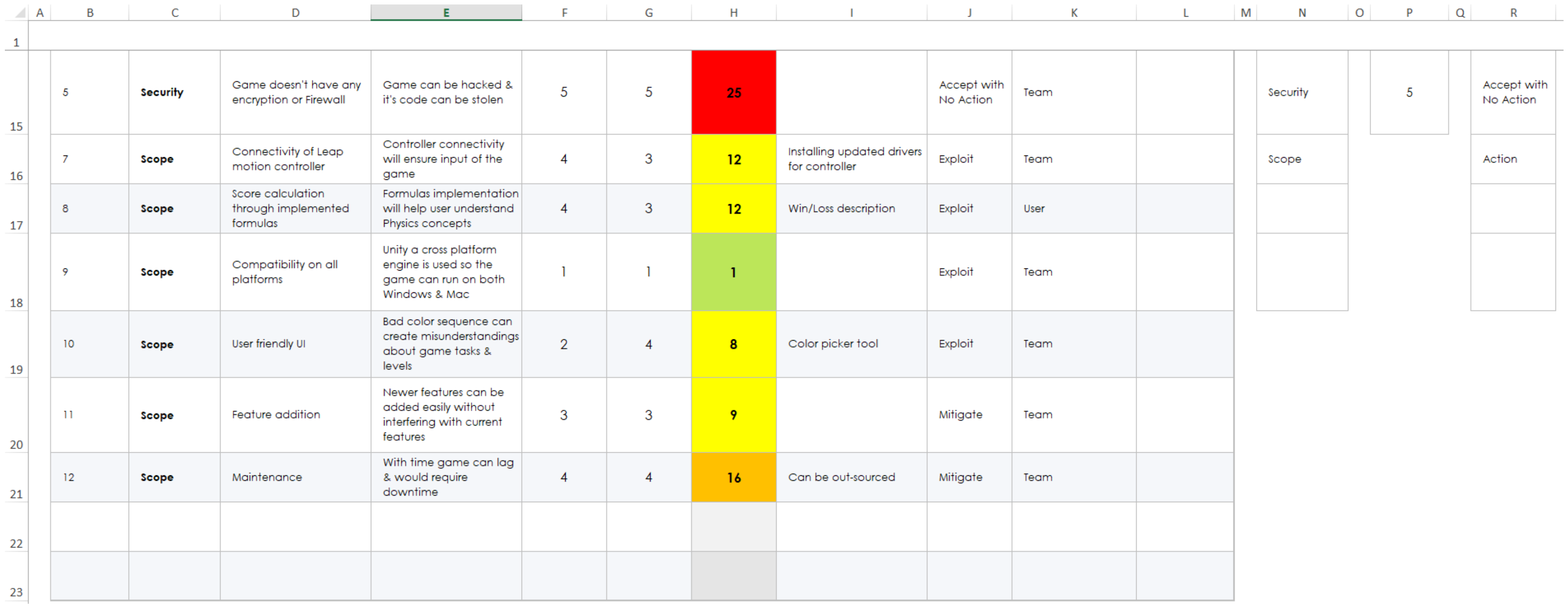
5	5	10	15	20	25
4	4	8	12	16	20
3	3	6	9	12	15
2	2	4	6	8	10
1	1	2	3	4	5
	1	2	3	4	5

IMPACT

RISK ID NO.	RISK CLASS	RISK DESCRIPTION	IMPACT DESCRIPTION	IMPACT LEVEL	PROBABILITY LEVEL	PRIORITY LEVEL	MITIGATION STRATEGY	ACTION	OWNER	REASSESSMENT DATE
	Risk Class Options	calculated risk and uncalculated risk	complete failure of project and too much delay	Rate 1 (LOW) to 5 (HIGH)	Rate 1 (LOW) to 5 (HIGH)	(IMPACT X PROBABILITY) Address the highest first.	out-source of exiting risk to eliminate the problem	Choose from the drop-down menu.	Team members	10/2/2021
1	**Timeline**	Complete the project within timeline and select the team to divide the project into them to accomplish the task	Bad reputation & chances to lose client	3	4	**12**	some part of the project can be out-sourced	Transfer / Share	Team members & third party source	
2	**Budget**	Cost Estimation	Providing right cost analysis to client will develop trust	1	2	**2**	Market cost survey	Exploit	Team members & client	
3	**Resources**	Building team for the project	Competitive & skilled developers will achieve the goal in prescribed time	1	1	**1**		Exploit	Team	
4	**Resources**	Gathering resources for project	Right resources will keep the project on track	1	1	**1**		Mitigate	Team and coordinator	

KEYS

RISK CLASS	LEVEL	ACTION
Solution	1	Avoid
Timeline	2	Exploit
Budget	3	Transfer / Share
Legislative	4	Mitigate



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1																		
15		5	**Security**	Game doesn't have any encryption or Firewall	Game can be hacked & it's code can be stolen	5	5	**25**		Accept with No Action	Team			Security		5		Accept with No Action
16		7	**Scope**	Connectivity of Leap motion controller	Controller connectivity will ensure input of the game	4	3	**12**	Installing updated drivers for controller	Exploit	Team			Scope				Action
17		8	**Scope**	Score calculation through implemented formulas	Formulas implementation will help user understand Physics concepts	4	3	**12**	Win/Loss description	Exploit	User							
18		9	**Scope**	Compatibility on all platforms	Unity a cross platform engine is used so the game can run on both Windows & Mac	1	1	**1**		Exploit	Team							
19		10	**Scope**	User friendly UI	Bad color sequence can create misunderstandings about game tasks & levels	2	4	**8**	Color picker tool	Exploit	Team							
20		11	**Scope**	Feature addition	Newer features can be added easily without interfering with current features	3	3	**9**		Mitigate	Team							
21		12	**Scope**	Maintenance	With time game can lag & would require downtime	4	4	**16**	Can be out-sourced	Mitigate	Team							
22																		
23																		

References

Non-Functional Requirements - Lawrence Chung, Department of Computer Science, University of Texas at Dallas

http://www.c-jump.com/CIS75/Week06/samples/requirements.html

https://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/

Structural Motion Grammar for Universal Use of Leap Motion: Amusement and Functional Contents Focused, Byungseok Lee, Donghwe Lee, and Seongah Chin, Division of Media Software, Sungkyul University, Anyang City, Republic of Korea

https://app.assembla.com/spaces/tank_wars/wiki

Product Line Use Cases: Scenario-Based Specification and Testing of Requirements, A. Bertolino, A. Fantechi, S. Gnesi, and G. Lami

UML and Patterns book

An Analysis of the Precision and Reliability of the Leap Motion Sensor and Its Suitability for Static and Dynamic Tracking by Jože Guna, Grega Jakus, Matevž Pogačnik, Sašo Tomažič and Jaka Sodnik Faculty of Electrical Engineering, University of Ljubljana, Tržaška 25, Ljubljana 1000, Slovenia

The Key Difference between Serious Games and Gamification in eLearning by COREAXIS

Glossary

Leap Motion: A computer hardware sensor device that supports hand and finger motions as input, but requires no hand contact or touching.

Unity3D: Unity is a cross-platform game engine developed by Unity Technologies

Dynamic objects: Moving objects in game, such as ball.

Static objects: Immovable objects, such as blocks.

Ground objects: Ground or walls of the game.