|  |  |  |
| --- | --- | --- |
| جـامعـــة الأزهــــر  كليـــــة الهندســـــة (بنين)  قسم هندسة النظم والحاسبات | Logo  Description automatically generated |  |
| Al-Azhar University |
| Faculty of Engineering (Boys) |
| Systems and Computer Engineering |
|  |

**Graduation Project**

**Logo

Description automatically generated**

**Project Name : Chemistry Virtual Lab**

**A Project Submitted in Partial Fulfillment**

**of the Requirements for the Degree of Bachelor of Science**

**in Systems and Computer Engineering**

**2023**

|  |  |  |
| --- | --- | --- |
| جـامعـــة الأزهــــر  كليـــــة الهندســـــة (بنين)  قسم هندسة النظم والحاسبات | Logo  Description automatically generated |  |
| Al-Azhar University |
| Faculty of Engineering (boys) |
| Systems and Computer Engineering |
|  |

**Graduation Project**

**Logo

Description automatically generated**

**Project Name : Chemistry Virtual Lab**

**A Project Submitted in Partial Fulfillment**

**of the Requirements for the Degree of Bachelor of Science**

**in Systems and Computer Engineering**

**Submitted By**

|  |
| --- |
| 1. **Omar AbdElnasser Ibrahime 404077** |
|  |
| **B-Yusuf Ashour Abdelrahman 404145** |
|  |
| **C-Ammar Mahmoud Mohamed 404074** |
|  |
| **D-Abdalla Ehab Salem Matar 404066** |
|  |
| **E-Yasser Ahmed Ahmed Issa 404139** |

**Supervised by**

**Prof. Dr. Shehab Gamal**

**2023**

# Examiner Committee

|  |  |  |
| --- | --- | --- |
| **Name** | **Rule** | **Signature** |
| Prof. Dr. Gamal Tharwat | Examiner |  |
| Prof. Dr. Shehab Gamal | Supervisor |  |
| Prof. Dr. Ashraf ElMarakby | Examiner |  |

**Date**

13st of July, 2023

# ABSTRACT

The Virtual Chemistry Lab project using Unity is a software application designed to provide students with a safe and accessible platform to conduct chemistry experiments and simulations. The project utilizes the Unity game engine to create a realistic and interactive virtual environment where students can explore and manipulate different variables and conditions to better understand chemical reactions and processes.

The application features a variety of experiments and simulations covering topics such as acid-base reactions, gas laws, and chemical equilibrium. The experiments are designed to be engaging and interactive, allowing students to perform actions such as mixing chemicals, adjusting temperature and pressure, and observing the resulting reactions and changes.

In addition to the experiments, the application also includes interactive tutorials and visualizations to help students better understand chemistry concepts and principles. The tutorials provide step-by-step instructions on how to perform experiments and explain the underlying chemical processes at work. The visualizations use 3D models and animations to illustrate complex concepts in a clear and concise manner.

KEYWORDS: Chemical Lab ; Experiment ; Unity ; C#

# ACKNOWLEDGEMENTS

First and foremost, we thank and praise **Allah** for providing us with the patience, strength, well-being, and skills to complete one of the most important stages in our lives.

We would like to thank our supervisor, ProfessorShehab Gamal, for his support and guidance.

We would also like to acknowledge Al-Azhar Universityfor granting us to be among its respectful students.

# TABLE OF CONTENTS

[ABSTRACT… iv](#_TOC_250084)

[ACKNOWLEDGEMENTS. v](#_TOC_250083)

[TABLE OF CONTENTS. vi](#_TOC_250082)

[LIST OF TABLES… xi](#_TOC_250081)

LIST OF FIGURES… xii

[GLOSSARY AND LIST OF ABBREVIATIONS… xiii](#_TOC_250080)

[CHAPTER 1 INTRODUCTION… 1](#_TOC_250079)

* 1. [Background 1](#_TOC_250078)
  2. [Motivation 2](#_TOC_250077)
  3. [Problem description 4](#_TOC_250076)
  4. [Aims and Objectives 5](#_TOC_250075)
  5. [Methodology 7](#_TOC_250074)

[CHAPTER 2 SYSTEM DESIGN… 21](#_TOC_250051)

* 1. Introduction………………………………………………………………….21
  2. [Requirements 21](#_TOC_250047)
     1. User requirements………………………………………………….21
     2. [System requirements 21](#_TOC_250045)
  3. [Process and Sequence Diagram 22](#_TOC_250044)
     1. [Use Case Diagram 22](#_TOC_250043)

2.3.2 [Sequence Diagrams 23](#_TOC_250042)

* 1. Database Design 27
     1. Entities with Attributes 27
     2. Relationships 28
     3. Entity-Relationship Diagram 29
     4. Database Schema 30
     5. Data Dictionary…………………………………………………….31

2.5Database Models 33

2.6 APIs Communication………………………………………………………………….35

[CHAPTER 3 IMPLEMENTATION… 36](#_TOC_250035)

* 1. [Introduction 36](#_TOC_250034)
  2. [Tools & Technologies 36](#_TOC_250033)
  3. [Preprocessing 38](#_TOC_250032)

3.4 Code 43

[CHAPTER 4 RESULTS AND DISCUSSION. 55](#_TOC_250013)

* 1. [Results Analysis and Discussion 55](#_TOC_250012)

4.2 [Testing Data 63](#_TOC_250009)

[CHAPTER 5 CONCLUSION AND FUTURE WORK… 69](#_TOC_250005)

* 1. [Conclusion 69](#_TOC_250004)

5.2 [Future Work 69](#_TOC_250003)

[REFERENCES… 70](#_TOC_250002)

# LIST OF TABLES

|  |  |
| --- | --- |
|  | Table1.1 Comparison between Virtual Labs And Practical Labs…………4 |
|  | Table1.2 Chemical Properties …………..………………………………..17 |
|  | Table1.3 Physical Properties………….…………………………………..18  Table2.1 Experiment Data Dictionary…………….…..…………….........31  Table2.2 Chemicals Data Dictionary……………………………………. 31  Table2.3 Steps Data Dictionary…………………………………………. 32  Table2.4 Equipment Data Dictionary………..……………………………32 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
|  | **LIST OF FIGURES** |
|  | Figure1.1 VR Photo………………………………………………….. 2 |
|  | Figure1.2 IKEA Place………………………………………………….3  Figure1.3 Cross-Platform Unity Engine ………....………………......10  Figure1.4 BEAKER ..………………………………………………….11  Figure1.5 GRADUATED CYLINDER ……………………………….12  Figure1.6 PASTEUR PIPETTE ....…………………………………….12  Figure1.7 TEST TUBE ….…………………………………………….13  Figure1.8 TEST TUBE RACK ….…………………………………….13  Figure1.9 BUNSEN BURNER .……………………………………….14  Figure1.10 Triangle Flame Holder ..…………………………………...14  Figure1.11 Periodic Table …...………………………………………...15  Figure1.12 Ionic Bond …………………………………………………16  Figure1.13 Covalent Bond ……………………..………………………17  Figure 2.1 Use Case Diagram………………………………………......22  Figure 2.2 Login Sequence Diagram…………………………………..23  Figure 2.3 Registration Sequence Diagram………………………...….24  Figure 2.4 Create Experiment Sequence Diagram……………………..25  Figure 2.5 Do Experiment Sequence diagram………………………….26  Figure 2.6 Experiment Entity…………………………………………...27  Figure 2.7 Equipment Entity…………………………………………....27  Figure 2.8 Chemical Entity……………………………………………...27  Figure 2.9 Steps Entity…………………………………………………..28  Figure 2.10 Relation 1………………………………………………….28  Figure 2.11 Relation 2………………………………………………….28  Figure 2.12 Relation 3………………………………………………….28  Figure 2.13 Entity Relationship Diagram ……………………………..29  Figure 2.14 Database Schema…………………………………………..30  Figure 2.15 API Communication1……………………………………...35  Figure 2.16 API Communication2……………………………………...35  Figure 3.1 Visual Studio…………………………………………………37  Figure 3.2 Unity Studio……………………………………………….....37  Figure 3.3 Unity Hub…………………………………………………….37  Figure 3.4 chemical Lab Room………………………………………….38  Figure 3.5 Safety Rules …………………………………………………39  Figure 3.6 Chemical Tools In Unity…………………………………….40  Figure 3.7 Choose Language Screen…………………………………….41  Figure 3.8 Unity UI Screens……………………………………………..42  Figure 4.1 First UI Screen……………………………………………….55  Figure 4.2 First Instruction………………………………………………56  Figure 4.3 Safety Rules In Game Mode…………………………………57  Figure 4.4 Second Instruction……………………………………………58  Figure 4.5 Third Instruction……………………………………………...59  Figure 4.6 Fourth Instruction…………………………………………….60  Figure 4.7 Fifth Instruction………………………………………………61  Figure 4.8 Sixth Instruction………………………………………………62  Figure 4.9 Test 1………………………………………………………….63  Figure 4.10 Test 2………………………………………………………64  Figure 4.11 Test 3………………………………………………………65  Figure 4.12 Test 4………………………………………………………66  Figure 4.13 Test 5………………………………………………………67  Figure 4.14 Test 6………………………………………………………68 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# GLOSSARY AND LIST OF ABBREVIATIONS

|  |  |
| --- | --- |
| **Abbreviation** | **Meaning** |
| VR | Virtual Reality |
| AR | Augmented Reality |
| V-Chemistry Lab | Virtual Chemistry Lab |
| UI | User Interface |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# CHAPTER 1 INTRODUCTION

This Documentation is about ‘V-**Chemistry Lab’** that help teachers and students to build and carry out chemical experiments so in this chapter will take about Background , Motivation examples , Problem Description that our project try to solve and Methodology to solve this problem.

## Background

Nowadays, E-learning has proven its importance in many fields, like Chemistry, Physics, Computer science, etc.

And the future will be for E-Learning where there is development in networks and computing devices.

As our project is virtual Chemistry Lab that is an application build with unity game engine this app will help student to carry out Chemical experiment with no need to go to Practical chemical Labs or Attend in school or university lab.

.

## Motivation

## 

## Ease of building and implementing the chemical experiment for students and teachers ,

## And there is many Motivation Examples Like:

## VR Application: a computer simulation that creates an image of a world that appears to our senses in much the same way we perceive the real world, or “physical ” reality.

## 

Figure 1.1 A virtual reality participant wearing a head-mounted display and a glove input

device interacts with a virtual world

## AR Applications : superimposes digital con-tent (text, images, animations, etc.) on a user’s view of the real world.

## 

Figure 1.2 IKEA Place

## Electrical circuit simulation like : Proteus

## And many other simulation program like : MatLab.

## All these applications help make life easier and simpler.

## Problem Description

Practical laboratory courses are an essential part of chemistry education. However, they can be costly and time-consuming. They also require physical presence of the teacher and students and access to well-equipped laboratories, which can be hindered due to equipment cost or a pandemic lockdown.

Table 1.1 Comparison between Virtual Labs And Practical Labs

|  |  |
| --- | --- |
| Virtual Labs | Practical Labs |
| Any number of students can attend | Limited number of students can attend |
| Expensive Tools | Virtual Tools |
| risk of injuring | No risk as all this is virtual |
| Developing the self-learning part for students | More Dependency on Teachers |
| More expensive | Less expensive as One single lab platform can serve an entire school |

## Aims and Objectives

The aim :

* To provide a safe and flexible learning environment for students to conduct experiments and explore chemical concepts.
* To enhance students' understanding of chemical principles and concepts through interactive simulations and virtual experiments.
* To promote critical thinking and problem-solving skills by presenting students with complex chemical problems and scenarios.
* To enable students to explore and experiment with different chemical reactions and processes without the limitations of a traditional laboratory setting.
* To provide teachers with tools to create custom experiments and simulations that align with their curriculum and learning objectives.

Objectives:

* To develop a user-friendly virtual platform that allows students to access and navigate virtual chemistry experiments and simulations.
* To create a diverse range of virtual experiments and simulations that cover key chemical concepts and principles.
* To ensure the accuracy and reliability of virtual experiments and simulations by incorporating real-world data and observations.
* To provide feedback and guidance to students as they conduct virtual experiments and simulations.
* To evaluate the effectiveness of virtual chemistry labs in enhancing students' understanding of chemical concepts and principles.
* To continuously update and improve the virtual chemistry lab platform to incorporate new technologies and features.

## Methodology

## Methodology include important parts as follow :

## Planning the project

## Target audience

## Choosing the development platform

## Collecting Data and Resources

## Implementation

## Test and refine

## Planning

## Define the scope and objectives of the project: This involves identifying

## the specific goals and objectives of the virtual chemistry lab project.

## It is important to have a clear understanding of the target audience,

## learning outcomes, and available resources.

## Conduct a needs assessment: This involves gathering information

## about the current state of chemistry education and identifying areas where a virtual chemistry lab can add value. This can also involve assessing the technological infrastructure and resources available for implementing the virtual lab.

## Determine the technical requirements: This involves identifying the hardware

## and software requirements for creating and implementing the virtual chemistry lab.

## This can include identifying the necessary computational resources,

## software development tools, and other technical requirements.

## Develop content and experiments: This involves creating the content and

## experiments that will be included in the virtual chemistry lab.

## This can involve developing custom experiments or adapting existing

## experiments to a virtual environment.

## Test and evaluate: This involves testing and evaluating the virtual chemistry lab

## to ensure that it is effective and meets the defined objectives.

## This can involve gathering feedback from stakeholders

## and users and making iterative improvements to the design.

## Implement and maintain: This involves implementing

## the virtual chemistry lab and ensuring that it is maintained and updated over time.

## This can include providing technical support, updating content and experiments,

## and monitoring usage and effectiveness.

## Target audience

## The target audience for a virtual chemistry lab project can vary depending on the educational level and learning objectives. Here are some examples of potential target audiences:

## 1. High school students: Virtual chemistry labs can be designed to supplement traditional classroom learning and provide students with a safe and engaging way to explore chemical concepts and conduct experiments. The virtual labs can be aligned with high school chemistry curricula and learning outcomes.

## 2. College students: Virtual chemistry labs can be used in college-level chemistry courses to provide students with a more flexible and convenient way to conduct experiments and explore chemical concepts. The virtual labs can be designed to align with specific college-level chemistry courses and learning outcomes.

## 3. Science educators: Virtual chemistry labs can be used by science educators to create custom experiments and simulations that align with their curriculum and learning objectives. The virtual labs can be designed to provide teachers with a flexible and cost-effective way to supplement traditional laboratory experiments.

## 4. Researchers: Virtual chemistry labs can be used by researchers to simulate and study chemical reactions and processes in a controlled and reproducible environment. The virtual labs can enable researchers to conduct experiments that may not be possible or practical in a traditional laboratory setting.

## Choosing the development platform

## how could we simulate each part of chemical Lab from building and moving 3D Object to make interactions between them, to achieve this task we need Game Engine that deals with 3d objects and Among the different game engines, the choice fell on Unity because :

## Beginner-friendly : Easy to learn and use.

## Cross-Platform Engine : More platforms supported

## Unity has a more large and active community

## Huge amount of assets ready to use

## Packages Support

## 6- using C# .

## 

## Figure 1.3 Cross-Platform Unity Engine

## Collecting Data and Resources

## As we discuss this application deals with chemical experiments so first we need to know what chemical experiment means and its Elements.

## Chemical experiment : science based on observation and experimentation. Doing chemistry involves attempting to answer questions and explain observations in terms of the laws and theories of chemistry.

## Elements of chemical experiment:

## Tools : each chemical experiment has its own tools that used to carry out this experiment.

## Each tool has its own function and there are some examples:

## BEAKER:

## 

## 

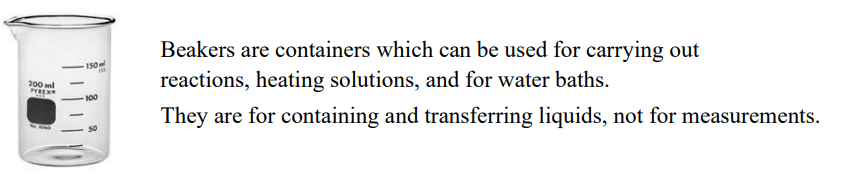


Figure 1.4 BEAKER

## GRADUATED CYLINDER

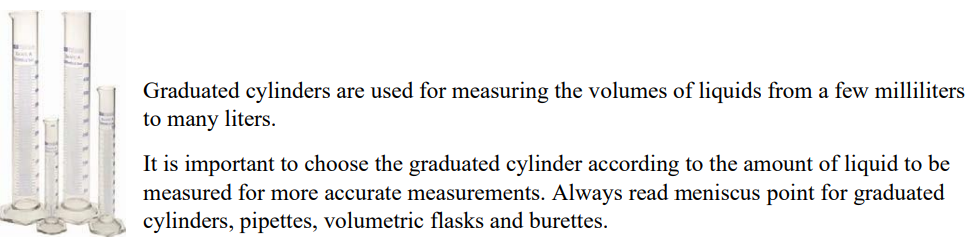


Figure1.5 GRADUATED CYLINDER

## PASTEUR PIPETTE

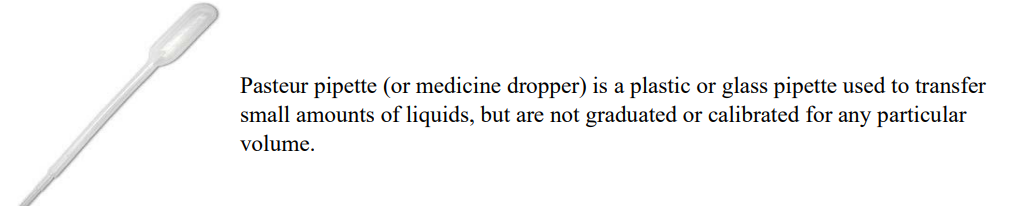
****

Figure ‎1.6 PASTEUR PIPETTE

## TEST TUBE

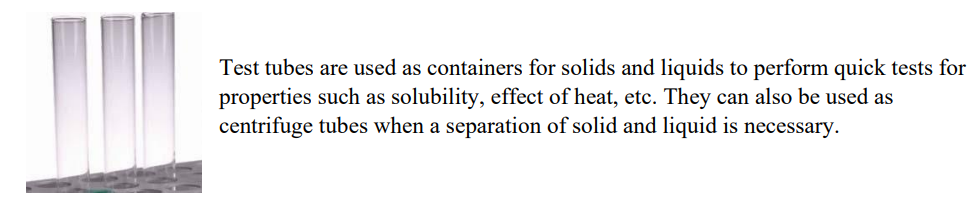
****

Figure 1.7 TEST TUBE

* TEST TUBE RACK

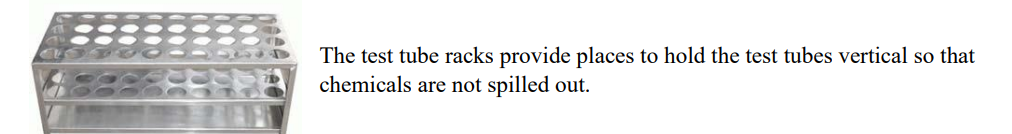


Figure 1.8 TEST TUBE RACK

* BUNSEN BURNER

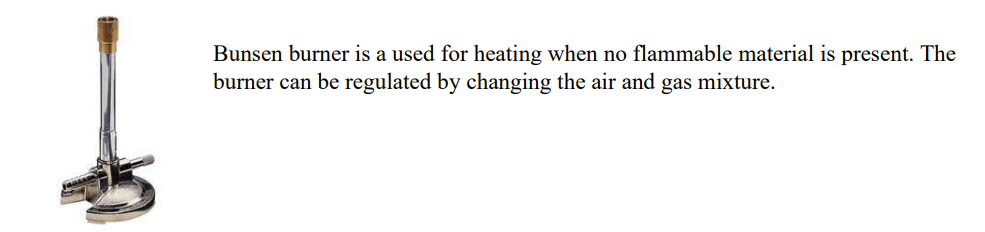


Figure 1.9 BUNSEN BURNER

* Triangle Flame Holder



Figure 1.10 Triangle Flame Holder

1. Chemical elements : a substance that can't be broken down by non-nuclear reactions.

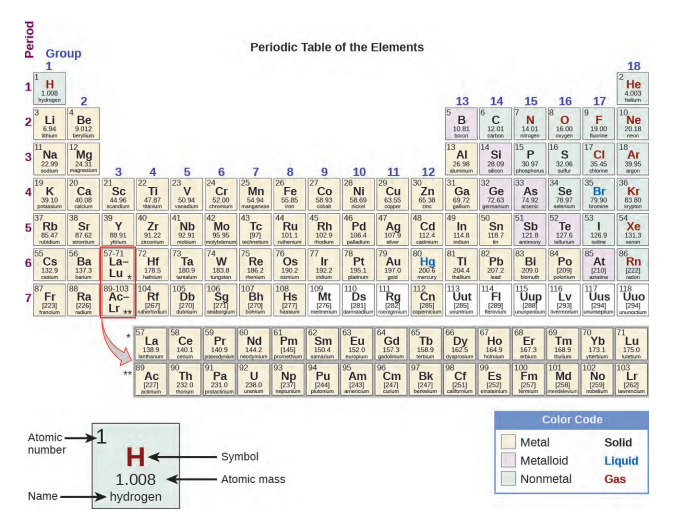


Figure 1.11 Periodic Table

1. chemical compounds: a chemical substance containing atoms from more than one chemical element held together by chemical bonds.

* Ionic bonding: formed between two or more atoms by the transfer of one or more electrons between atoms.

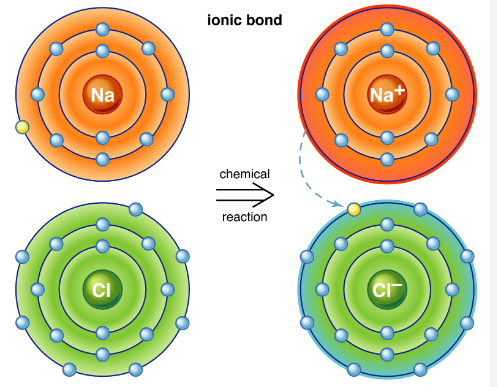


Figure ‎1.12 Ionic Bond

* Covalent bond: involves the sharing of electrons to form electron pairs between atoms

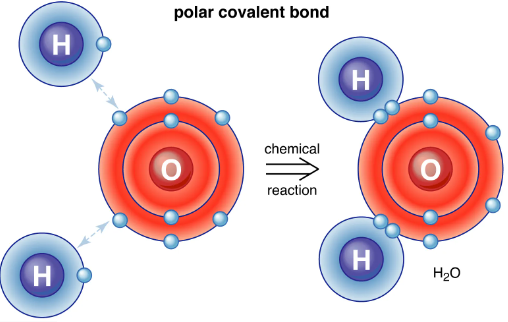


Figure 1.13 Covalent Bond

## Each of the chemical elements and compounds has what distinguishes it from others, such as the chemical name , symbol, and it also has physical properties and chemical properties that must be taken care of as much as possible when simulating.

## Chemical properties:

Table 1.2 Chemical Properties

|  |  |  |
| --- | --- | --- |
| Atomic number | indicates the number of protons within the core of an atom. | Na = 11 |
| Chemical Valence | number of electrons that an atom loses, gains, or shares during a chemical reaction | Na loss its electron to be positive ion (Na+) |
| Electrons per shell. | How electrons will distributed throw each shell | Na = 2,8,1 |

* physical properties :

Table 1.3 Physical Properties

|  |  |
| --- | --- |
| Property Name | Na |
| Melting point | 97.72 °C |
| Boiling point | 882.940 °C |
| Color | Silvery-White |
| State of Matter | Solid |
| Electrical Conducting | Good Conductor |
| Thermal conducting | Good Conductor |

last thing we need to take care about in chemical part is that each chemical experiment has :

1. steps to achieve this experiment and some experiment require pre-request steps
2. The observation we take from a chemical experiment
3. Conclusions

**Implementation**

To create a virtual chemistry lab in Unity using C#, there are several steps that we need to follow:

1. First, we need to create a new project in Unity and import any necessary assets, such as models, textures, or audio files.

2. Next, we need to design the user interface for the virtual chemistry lab using Unity's built-in UI tools.

3. After that, we need to create the virtual lab environment, including the laboratory equipment, chemicals, and other objects, using Unity's 3D modeling tools.

4. We also need to write the code to implement the chemistry experiments in the virtual lab, which can involve creating scripts in C# that simulate chemical reactions and other processes.

5. Additionally, we need to write the code to enable user interaction with the virtual objects and experiments, such as mouse clicks or keyboard input.

6. To enhance the user experience, we should add audio and visual effects to the virtual chemistry lab, such as sound effects for chemical reactions or particle effects for chemical processes.

7. Finally, we need to test the virtual chemistry lab and refine the design and implementation as necessary, gathering feedback from stakeholders and users to identify areas for improvement.

**Test and refine**

Testing and refining are important steps in the development of a virtual chemistry lab project. Here are some steps that can be included in the testing and refining phase:

1. Conduct usability testing: This involves testing the virtual chemistry lab with a group of representative users to identify any usability issues or challenges. Usability testing can provide valuable feedback on the design of the virtual lab and help to identify areas for improvement.

2. Gather feedback from stakeholders: This involves gathering feedback from stakeholders, such as teachers, students, and administrators, to understand their perspectives on the virtual chemistry lab. Feedback can be gathered through surveys, focus groups, or interviews.

3. Evaluate learning outcomes: This involves evaluating the impact of the virtual chemistry lab on learning outcomes, such as student engagement, understanding of chemical concepts, and performance on assessments. Evaluation can be conducted through pre- and post-tests, surveys, or other assessment methods.

4. Make iterative improvements: Based on the feedback and data gathered from testing and evaluation, make iterative improvements to the virtual chemistry lab. This can involve refining the design, improving the content and experiments, or addressing technical issues.

5. Test again: After implementing improvements, conduct additional testing to ensure that the virtual chemistry lab is effective and meets the defined objectives. This can involve repeating usability testing, gathering additional feedback from stakeholders, and evaluating learning outcomes again.

# CHAPTER 2

# SYSTEM DESIGN

## Introduction

## In this chapter we will take a look on database, APIs , User and system requirement

## And finally database model.

## Requirements

## 2.2.1 User requirements

## Application has two point of view:

## Teacher or supervisor should be able to:

## Create chemical experiment by choosing its Name , Tools , compounds,

## Steps and conclusion.

## Student should be able to :

## Carry out experiment

## Learn from its mistakes

## 2.2.2 System requirements

## SQL Lite database

## Game Engine(Unity) to deal with 3D Objects.

## Process and Sequence Diagram

## 2.3.1 Use Case Diagram

## 

Figure 2.1 Use Case Diagram

## 2.3.2 Sequence Diagram

* Login



Figure 2.2 Login Sequence Diagram

* Registration

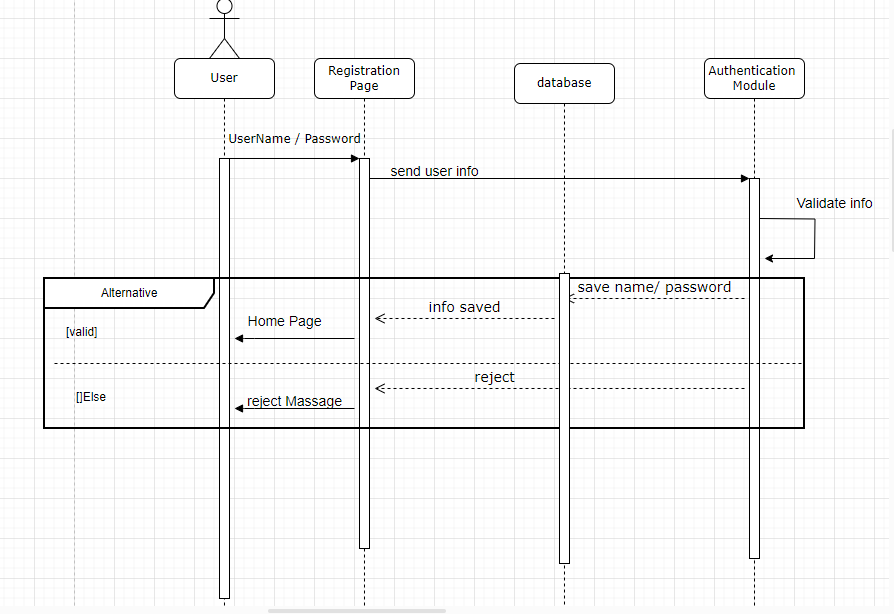


Figure 2.3 Registration Sequence Diagram

* Create Experiment

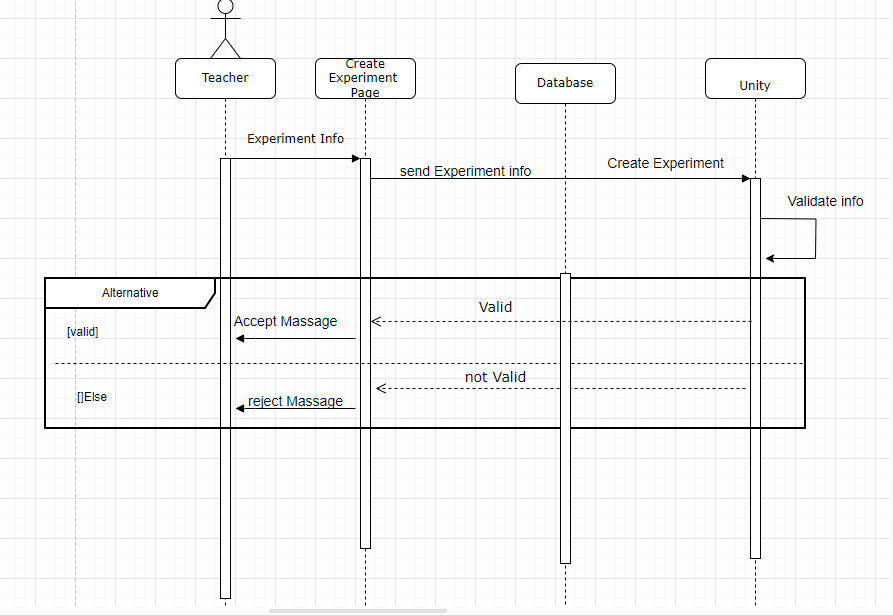


Figure 2.4 Create Experiment Sequence Diagram

* Do Experiment



Figure 2.5 Do Experiment Sequence diagram

## Database Design

## 2.4.1 Entities with Attributes

## Experiment

## 

Figure 2.6 Experiment Entity

## 

* Equipment

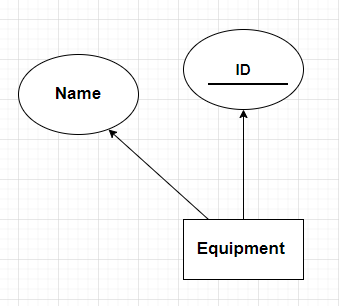


Figure 2.7 Equipment Entity

* Chemicals

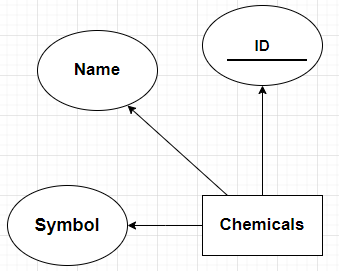


Figure 2.8 Chemical Entity

* Steps

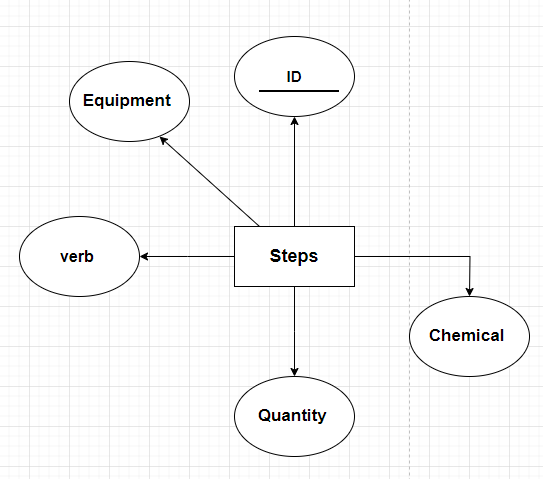


Figure 2.9 Steps Entity

## 2.4.2 Relationships

## Belong To

## 

Figure 2.10 Relation 1

* Used In



Figure 2.11 Relation 2

* Has



Figure 2.12 Relation 3

## 2.4.3 Entity-Relationship Diagram

## 

Figure 2.13 Entity Relationship Diagram

## 2.4.4 Database Schema

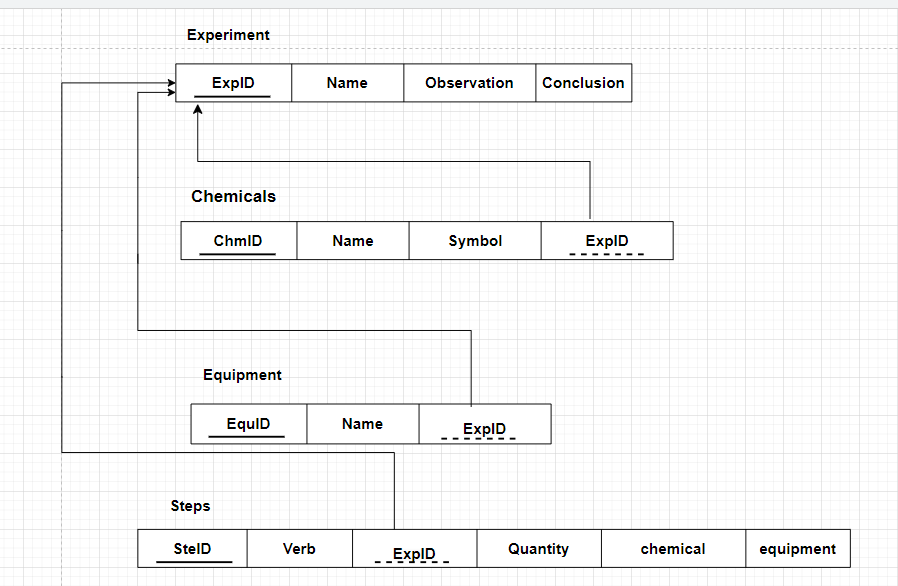


Figure 2.14 Database Schema

## 2.4.5 Data Dictionary

## Experiment

Table 2.1 Experiment Data Dictionary

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| NO | Attribute Name | Data Type | Description | PK | FK | Reference |
| 1 | ExpID | Number(3) |  | ✓ |  |  |
| 2 | Name | Varchar2(20) | Not Null |  |  |  |
| 3 | Observation | Varchar2(100) | Not Null |  |  |  |
| 4 | conclusion | Varchar2(100) | Not Null |  |  |  |

* Chemicals

Table 2.2 Chemicals Data Dictionary

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| NO | Attribute Name | Data Type | Description | PK | FK | Reference |
| 1 | ChmID | Number(3) |  | ✓ |  |  |
| 2 | Name | Varchar2(20) | Not Null |  |  |  |
| 3 | Symbol | Varchar2(3) | Not Null |  |  |  |
| 4 | ExpID | Number**(3)** |  |  | **✓** | Experiment |

* Steps

Table 2.3 Steps Data Dictionary

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| NO | Attribute Name | Data Type | Description | PK | FK | Reference |
| 1 | SteID | Number(3) |  | ✓ |  |  |
| 2 | Verb | Varchar2(8) | Not Null |  |  |  |
| 3 | Quantity | Number(2) | Not Null |  |  |  |
| 4 | chemical | Varchar2(100) | Not Null |  |  |  |
| 5 | equipment | Varchar2(100) | Not Null |  |  |  |
| 6 | ExpID | Number**(3)** |  |  | **✓** | Experiment |

* Equipment

Table 2.4 Equipment Data Dictionary

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| NO | Attribute Name | Data Type | Description | PK | FK | Reference |
| 1 | EquID | Number(3) |  | ✓ |  |  |
| 2 | Name | Varchar2(20) | Not Null |  |  |  |
| 3 | ExpID | Number**(3)** |  |  | **✓** | Experiment |

## Database Models

|  |
| --- |
| from django.db import models  class Experiment(models.Model):  name = models.CharField(max\_length=1000)  observation = models.CharField(max\_length=1000)  conclusion = models.CharField(max\_length=1000)    def \_\_str\_\_(self) -> str:  return self.name  class Equipment(models.Model):  EQUIPMENT\_CHOICES = (  ('Beaker', 'Beaker'),  ('Bunsen Burner', 'Bunsen Burner'),  ('Sunflower Paper', 'Sunflower Paper'),  ('Pipette', 'Pipette'),  ('Test Tube', 'Test Tube'),  ('Lighter', 'Lighter'),  ('Test Tube', 'Test Tube')  # add more choices here    )  experiment = models.ForeignKey(Experiment, on\_delete=models.CASCADE,related\_name='equipments')  name = models.CharField(max\_length=1000, choices=EQUIPMENT\_CHOICES)    def \_\_str\_\_(self) -> str:  return self.name  class Chemicals(models.Model):  CHEMICALS\_CHOICES = (  ('Water', 'Water'),  ('Starch', 'Starch'),  ('Sodium Hydroxide', 'Sodium Hydroxide'),  # add more choices here    )  experiment = models.ForeignKey(Experiment, on\_delete=models.CASCADE,related\_name='chemicals')  name = models.CharField(max\_length=1000, choices=CHEMICALS\_CHOICES)  symbol = models.CharField(max\_length=1000)    def \_\_str\_\_(self) -> str:  return self.name  class Steps(models.Model):  VERBS = (  ('Boil', 'Boil'),  ('Add', 'Add'),  ('Light', 'Light'),  # add more choices here  )  EQUIPMENT = (  ('Beaker', 'Beaker'),  ('Bunsen Burner', 'Bunsen Burner'),  ('Sunflower Paper', 'Sunflower Paper'),  ('Pipette', 'Pipette'),  ('Test Tube', 'Test Tube'),  ('Lighter', 'Lighter'),  ('Test Tube', 'Test Tube')  # add more choices here  )    CHEMICALS = (  ('Water', 'Water'),  ('Starch', 'Starch'),  ('Sodium Hydroxide', 'Sodium Hydroxide'),  # add more choices here  )  experiment = models.ForeignKey(Experiment, on\_delete=models.CASCADE, related\_name='steps')  verb = models.CharField(max\_length=100, choices=VERBS)  quantity = models.IntegerField(blank=True, null=True)  chemical = models.CharField(max\_length=100, choices=CHEMICALS, blank=True, null=True)  equipment = models.CharField(max\_length=100, choices=EQUIPMENT, blank=True, null=True)  def \_\_str\_\_(self) -> str:  if not self.quantity:  self.quantity = ''  if not self.chemical:  self.chemical = ''  if not self.equipment:  self.equipment = ''  return f'{self.verb} {self.quantity} {self.chemical} {self.equipment}'    @property  def formatted\_step(self) -> str:  if not self.quantity:  self.quantity = ''  else:  self.quantity = f'{self.quantity} ml'  if not self.chemical:  self.chemical = ''  if not self.equipment:  self.equipment = ''  if self.equipment and self.chemical:  self.equipment = f'to {self.equipment}'  self.chemical = f'from {self.chemical}'  return f'{self.verb} {self.quantity} {self.chemical} {self.equipment}' |

## APIs Communication

## 

Figure 2.15 API Communication1

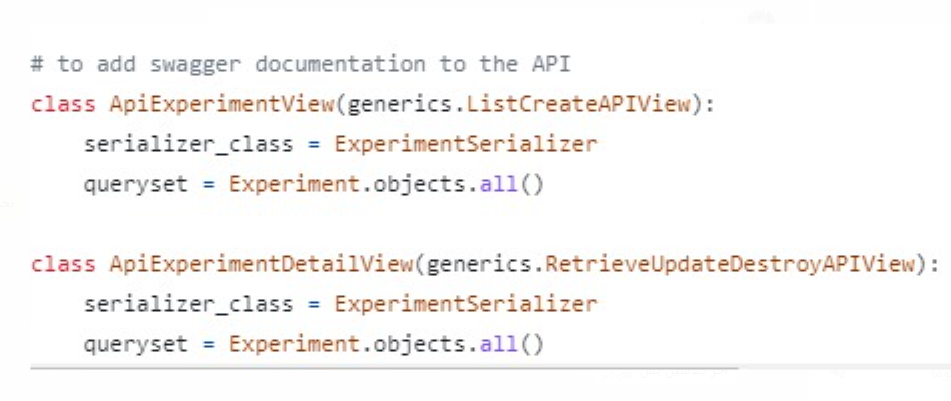


Figure 2.16 API Communication2

# CHAPTER 3 IMPLEMENTATION

## 3.1 Introduction

## For this chapter we are going to talk about tools and technology that used to build

## this project , also we will talk about Preprocessing part to see the 3D,2Dobjects

## that we need to carry out this chemical experiment and finally we will show code that

## help us to do interactions between this different objects.

## 3.2 Tools & Technologies

## Tools used in project:

## Visual studio

## Unity Game Engine

## Unity Hub

## Programing Language :

## C#

Tools and technologies brief :

## Visual studio : is an integrated development environment (IDE) from Microsoft,

## It is used to develop computer programs including websites, web apps,

## web services , mobile apps and games.

## 

Figure 3.1 Visual Studio

* Unity Game Engine:Unity is a cross-platform game engine can be used to create three-dimensional (3D) and two-dimensional (2D) games, as well as interactive simulations .



Figure 3.2 Unity Studio

## Unity Hub : is a standalone application that streamlines the way you find, download, and manage your Unity Projects and installations. In addition, you can manually add versions of the Editor that you have already installed on your machine to your Hub

## 

Figure 3.3 Unity Hub

## 3.3 Preprocessing

## Unity helps in control of the basic 3D shapes like : Cube and sphere to build:

## Room for our chemical experiments this Room consist of :

## Tables, Walls , Celling, Floor, Doors and source of light As shown in figure:

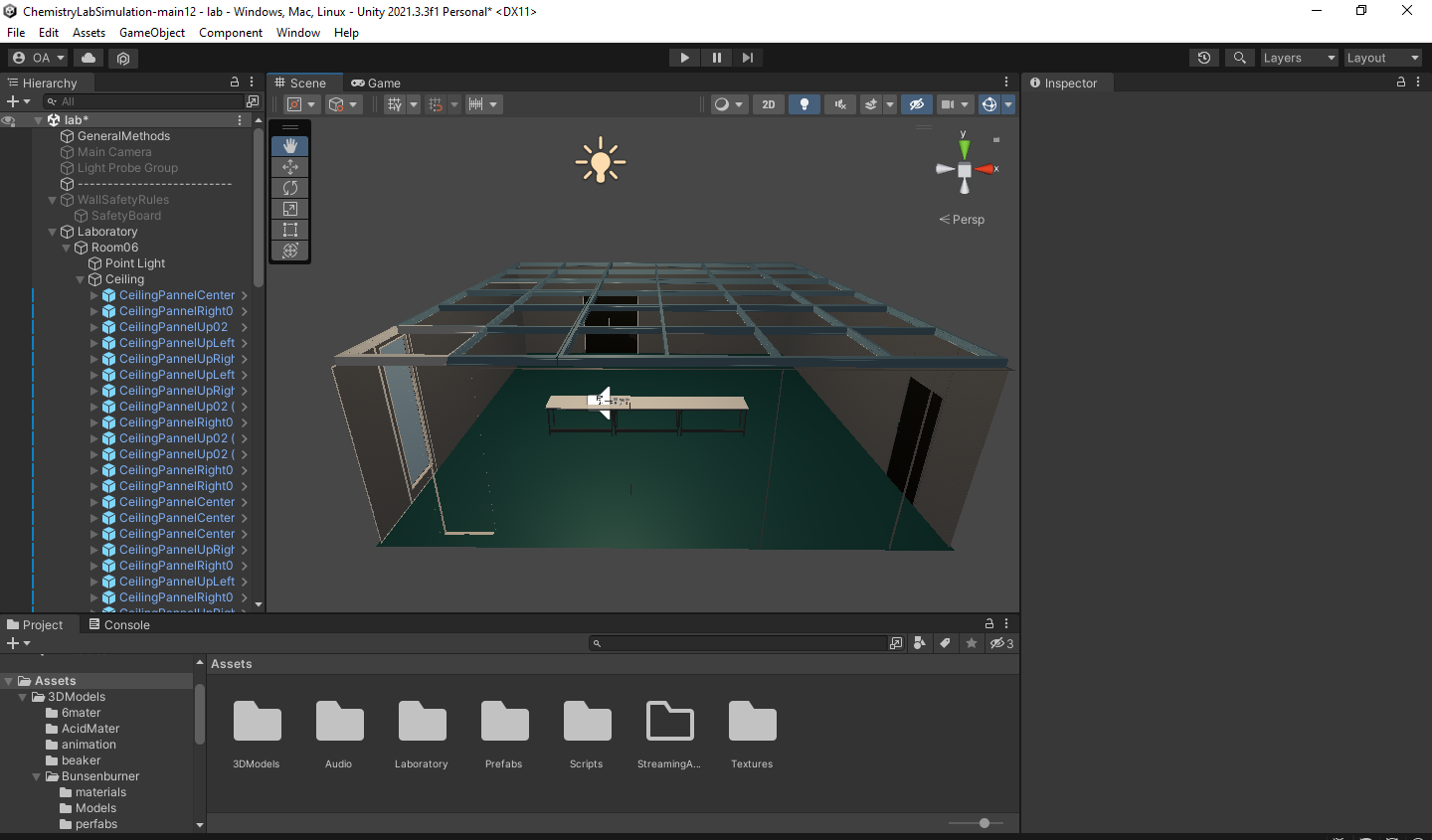


Figure 3.4 chemical Lab Room

## safety wall



Figure 3.5 Safety Rules

## After that we need chemical Tools which is not basic 3DObjects so these tools

## can be brought from :

## programs that creating 3DObjects like Blender or Autodesk Maya

## Unity Assets store: home to a growing library of free and commercial 3DObjects

## created by other developer and this is best choice to save time and effort.

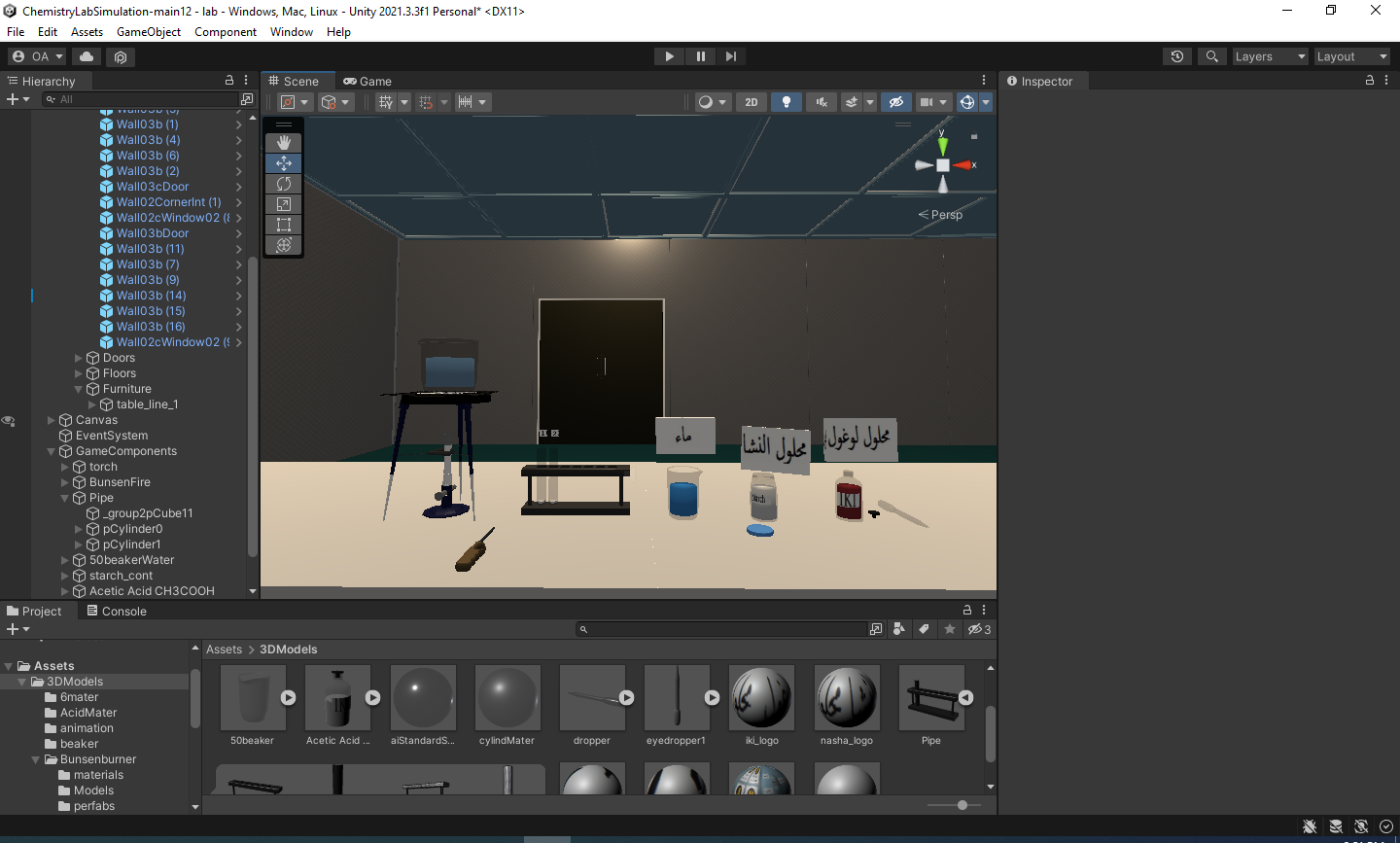
****

Figure 3.6 Chemical Tools In Unity

## Last thing we need to talk about in this section is Unity UI that you can use

## to create user interfaces (UI) for the Unity Editor and applications made in

## the Unity Editor , we use that UI in two places :

## First screen to choose language(English , Arabic)

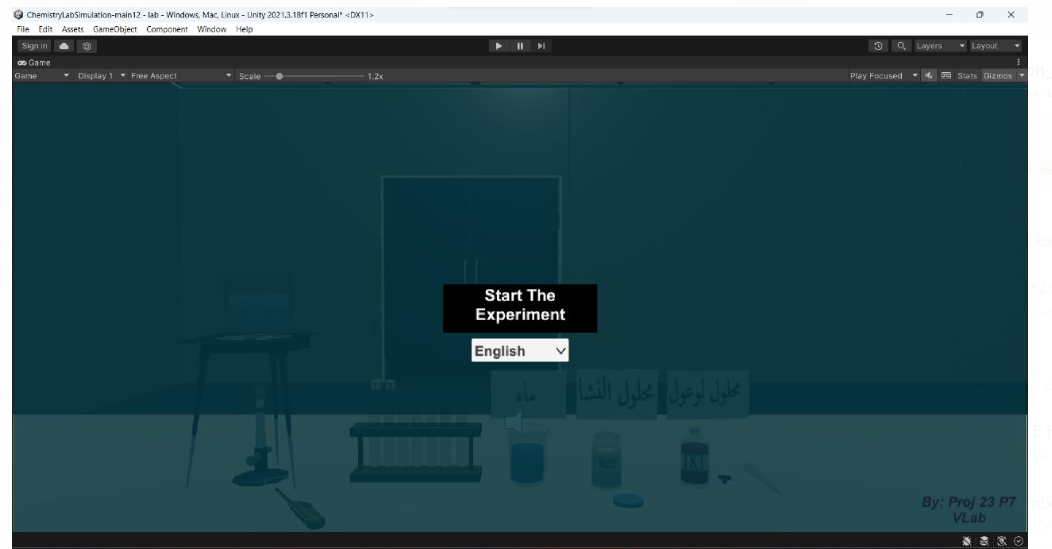


Figure 3.7 Choose Language Screen

* Second place in Time screen , Amount screen, Instructions screen and

Wrong screen As shown :

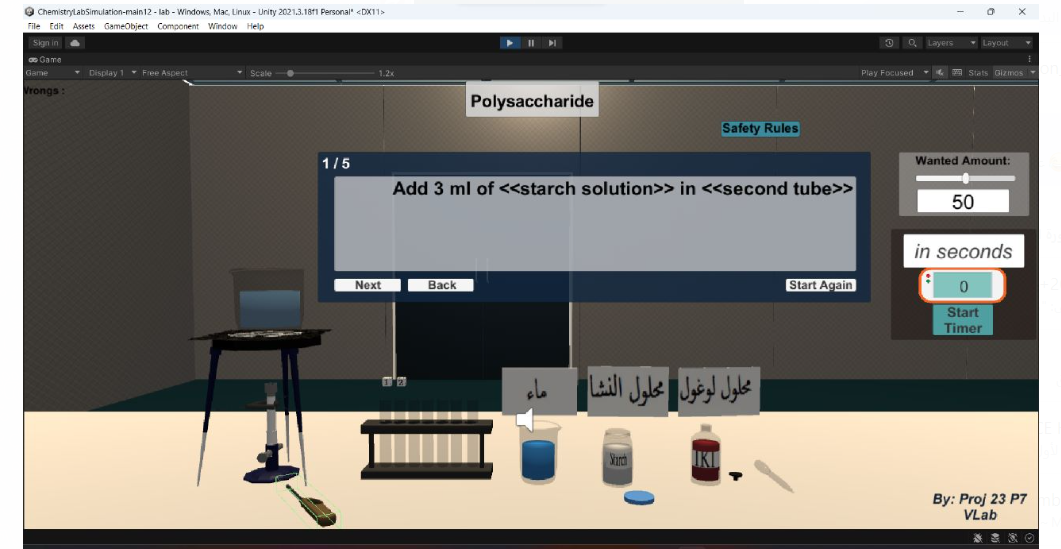


Figure 3.8 Unity UI Screens

## 3.4 Code

## By C# Scripts that will be put on 3D object that shown previous

## we can control its behavior and how it will interact with other

## 3D Object in the scene .

## GeneralMethods script on (General Methods Game Object) :

|  |
| --- |
| **using** **System**;  **using** **System.Collections**;  **using** **System.Collections.Generic**;  **using** **UnityEngine**;  **using** **UnityEngine.UI**;  **using** **Newtonsoft.Json**;  **using** **System.Net.Http**;  **using** **System.Threading.Tasks**;  **namespace** **ChemistryLab**  {**public** **static** **class** **APIHandler**{  **public** **static** **async** Task<List<ResponseObject>> FetchExperimentData()  {  HttpClient client = **new** HttpClient();  List<ResponseObject> responseObjects = **new** List<ResponseObject>();  **try**{  HttpResponseMessage response = **await** client.GetAsync("http://24.199.120.99/api/experiments/");  // Check if the request was successful  **if** (response.IsSuccessStatusCode)  {  **string** jsonResponse = **await** response.Content.ReadAsStringAsync();  // Deserialize JSON response into a list of objects  responseObjects = JsonConvert.DeserializeObject<List<ResponseObject>>(jsonResponse);  }  **else**  {  Console.WriteLine("Error occurred: " + response.StatusCode);  }}**catch** (Exception ex)  {  Console.WriteLine($"An error occurred: {ex.Message}");  }  **finally**  {  // Dispose of the HttpClient instance when done  client.Dispose();}  **return** responseObjects; }} |

|  |
| --- |
| **public** **class** **ResponseObject**  {  **public** **int** Id { **get**; **set**; }  **public** **string** Name { **get**; **set**; }  **public** List<Equipment> Equipments { **get**; **set**; }  **public** List<Chemical> Chemicals { **get**; **set**; }  **public** List<Step> Steps { **get**; **set**; }  **public** **string** Observation { **get**; **set**; }  **public** **string** Conclusion { **get**; **set**; }  }  **public** **class** **Equipment**  {  **public** **string** Name { **get**; **set**; }  }  **public** **class** **Chemical**  {  **public** **string** Name { **get**; **set**; }  }  **public** **class** **Step**  {  **public** **string** Verb { **get**; **set**; }  **public** **string** Equipment { **get**; **set**; }  **public** **string** Chemical { **get**; **set**; }  **public** **string** Quantity { **get**; **set**; }  }  **public** **class** **GeneralMethods** : MonoBehaviour  {  **public** Text instructionText;  **public** **static** GeneralMethods sing;//singelton pattern  **public** SkinnedMeshRenderer Pipe2FutureStarchSkRndrer;  //observer pattern  **public** **delegate** **void** **MyObserverDelegete**();  **public** MyObserverDelegete myMissionEventObserver;  **public** MyObserverDelegete myAlertsEventObserver;  **public** MyObserverDelegete ResetNowObserver;  **public** **bool** TimerEnabled = **false**;  **private** **float** startWatch;  **bool** TimeEnds = **false**;  **public** **void** **EnableTimer**()  {  **if** (toWaitTime != **12**)  {  **if** (ParameterFixingGlitchWaitTime == **0**)  {  UIObjsHandler.instance.AlertString = "Wrong , Set Timer time first";  }  **else**  {UIObjsHandler.instance.AlertString = "Wrong timer set , correct it with 120";  }  **return**;}  **if** (TasksManager.MissionID < **7**)  {  UIObjsHandler.instance.AlertString = "Finish previous nessecessary tasks first";  **return**;  }  **if** (TasksManager.MissionID == **7**)  {  **if** (FireBunsenHandler.singelton.myfireState != FireBunsenHandler.FireState.InFire)  {  UIObjsHandler.instance.AlertString = "Kidding? Turn The Fire On First";  **return**;  }  }  TimeEnds = **false**;  TimerEnabled = **true**;  startWatch = Time.time;  }  **float** ParameterFixingGlitchWaitTime = **0**;//if using toWaitTime directly ,, (out) will make a glitch where any num is accepted while Timer already running  **public** **void** **OnEndEdit**(**string** time)  {  **if** (**float**.TryParse(time, **out** ParameterFixingGlitchWaitTime))  {  **if** (ParameterFixingGlitchWaitTime != **120**)  {  UIObjsHandler.instance.AlertString = "Wrong , 2 minutes required";  }  **else**  {  toWaitTime = ParameterFixingGlitchWaitTime / **10**; //faking the time for not let user wait so long  }  }  **else**  {  UIObjsHandler.instance.AlertString = "Wrong , Enter valid seconds";  }  }  **float** spentTime;  **float** toWaitTime = **0**;  **private** **void** **Update**()  {  **if** (TimerEnabled)  {  spentTime = Time.time - startWatch;  **if** (spentTime >= toWaitTime)  {  TimeEnds = **true**;  TimerEnabled = **false**;//must  **if** (TasksManager.MissionID == **7**)  {  //change nasha color to Purble now  **if** (Pipe2FutureStarchSkRndrer)  Pipe2FutureStarchSkRndrer.material.color = **new** Color32(**143**, **0**, **254**, **255**);  TasksManager.MissionID++;  }  }  **else**  {  UIObjsHandler.instance.TimerTxt.text = (spentTime \* **10f**).ToString();//faking 12 seconds to be 120  }  }  }  **public** **void** **OnValueChanged**(**float** val)  {  UIObjsHandler.instance.amountofSolInputSTR = val.ToString(); // this is a proberty that affects mutlible elemnts  }  **public** **void** **OnValueChanged**(Int32 chosenlang)  {  SetEnglishInstructions();  }  **void** **SetEnglishInstructions**()  {  **if** (instructionText)  {  instructionText.alignment = TextAnchor.MiddleLeft;  instructionText.lineSpacing = **1**;  }  }  **private** **async** **void** **Awake**()  {  sing = **this**;  // Fetch experiment data from the API  List<ResponseObject> responseObjects = **await** APIHandler.FetchExperimentData();  **if** (responseObjects.Count > **0**)  {  ResponseObject firstObject = responseObjects[**0**];  List<Equipment> equipments = firstObject.Equipments;  List<Step> steps = firstObject.Steps;  **string** observation = firstObject.Observation;  **string** conclusion = firstObject.Conclusion;  // Set the instructions based on the fetched data  TheInstructions = **new** **string**[steps.Count + **4**];  SetEnglishInstructions();  **string** all\_equipment = "";  **for**(**int** i = **0**; i < equipments.Count; i++)  {  all\_equipment += equipments[i].Name + ", ";  }  TheInstructions[**0**] = " ";  TheInstructions[**1**] = "See safety instructions (click the button at the top)";    **for** (**int** i = **0**; i < steps.Count; i++)  {  **if**(steps[i].Verb == "Add"){  steps[i].Quantity = " " + steps[i].Quantity + " ml from ";  steps[i].Equipment = " to " + steps[i].Equipment;  }  **else** **if**(steps[i].Verb == "Light"){  steps[i].Verb = "Light the ";  steps[i].Chemical = "";  steps[i].Quantity = "";  }  **else** **if**(steps[i].Verb == "Boil"){  steps[i].Verb = "Boil ";  steps[i].Equipment = "";  steps[i].Quantity = "";  }  **if**(i == **3**){  TheInstructions[**3**] = "Tools Needed: "+ all\_equipment+"\n--------\n "+steps[i-**2**].Verb + steps[i-**2**].Quantity + steps[i-**2**].Chemical + steps[i-**2**].Equipment;  }      TheInstructions[i+**2**] = steps[i].Verb + steps[i].Quantity + steps[i].Chemical + steps[i].Equipment;  }  **int** x = steps.Count+**2**;  TheInstructions[x] = "Observation: "+observation;  TheInstructions[x+**1**] = "Conclusion: "+conclusion; }  }  **void** **Start**()  {  ResetNowObserver += ResetGeneral;  //missionPassed observer  myMissionEventObserver += GoToMissionInstruction;  }  **public** **string**[] TheInstructions;  **public** **void** **NextInstruction**()  {  UIObjsHandler.instance.StepTxtDeal++;  UIObjsHandler.instance.TxtOfInstr = TheInstructions[UIObjsHandler.instance.StepTxtDeal];  }  **public** **void** **GoToMissionInstruction**()  {  UIObjsHandler.instance.StepTxtDeal = TasksManager.MissionID;  UIObjsHandler.instance.TxtOfInstr = TheInstructions[TasksManager.MissionID];  }  **public** **void** **PrevInstruction**()  {  UIObjsHandler.instance.StepTxtDeal--;  UIObjsHandler.instance.TxtOfInstr = TheInstructions[UIObjsHandler.instance.StepTxtDeal];  }  **public** **void** **ToggleInstructionPanel**()  {  UIObjsHandler.instance.InstructionPanel.SetActive(!UIObjsHandler.instance.InstructionPanel.activeInHierarchy);  UIObjsHandler.instance.ChooseAmountPanel.SetActive(!UIObjsHandler.instance.ChooseAmountPanel.activeInHierarchy);  }  **public** **void** **ResetGeneral**()  {  TimerEnabled = **false**;  toWaitTime = **0**;  UIObjsHandler.instance.TimerTxt.text = "0.0";  }  **private** **void** **OnDisable**()  {  myMissionEventObserver -= GoToMissionInstruction;  ResetNowObserver -= ResetGeneral;}}} |

* BlendInfo script on (Pipe => pCylinder => Blend):

|  |
| --- |
| **using** **System.Collections**;  **using** **System.Collections.Generic**;  **using** **UnityEngine**;  **namespace** **ChemistryLab**  {  **public** **class** **BlendInfo** : MonoBehaviour  {  **public** **bool** water = **false**;  **public** **bool** starch = **false**;  **public** **bool** IKI = **false**;  **public** SkinnedMeshRenderer skMsRndr ;  **public** Material material ;  **public** Color materialblendColor;  **void** **Awake**()  {  **if**(!skMsRndr)  skMsRndr = transform.GetComponent<SkinnedMeshRenderer>();  **if**(skMsRndr)  material = skMsRndr.material;  materialblendColor = material.color;}}} |

* DropperManager script on (Water\_dropper Game Object)

|  |
| --- |
| **using** **System.Collections**;  **using** **System.Collections.Generic**;  **using** **UnityEngine**;  **namespace** **ChemistryLab**  {  **public** **class** **DropperManager** : Movables  {  **public** **void** **ResetDroppersSitu**()  {  prevAmountOfSolinthisDrpr = **0f**;  AmountDrpNeeds = **0f**;  Destroy(oldBlendClone); //empty at first  }  **public** **void** **OnEnable**()  {  GeneralMethods.sing.ResetNowObserver += ResetDroppersSitu;  }  **private** **void** **OnDisable**()  {  GeneralMethods.sing.ResetNowObserver -= ResetDroppersSitu;  }  **public** **static** **float** ChoosenAmountOfSolute = **50f**; //any choosen value is global for all droppers so its static  //[SerializeField] GameObject Blend;  **public** Transform BlendTPos;  SkinnedMeshRenderer thisDropperBlClone\_skMeshRderer;  [SerializeField] **bool** isEmpty = **true**;  Quaternion initialRotation, inActionRotation = **new** Quaternion(**0**, **90**, **90**, **0**);  **public** **override** **void** **MakeAnotherSeperatedAction**()  {  RotateTheDropper();  }  **public** **void** **RotateTheDropper**()  {  //thisTransform.Rotate(thisTransform.position, degree);  thisTransform.localRotation = inActionRotation;  }  **private** **void** **FixedUpdate**()  {  }  **protected** **override** **void** **Awake**()  {  **base**.Awake();  initialRotation = thisTransform.localRotation;  }  **protected** **override** **void** **OnMouseUp**()  {  **base**.OnMouseUp(); // a must in c#  thisTransform.localRotation = initialRotation;  }  **bool** thisDropperIsAbsorbinSol;// ( out ) refrence , getting with the solutuion\_interact method  **private** **float** prevAmountOfSolinthisDrpr = **0f**;  **float** AmountDrpNeeds;  **private** **void** **OnTriggerEnter**(Collider other)  { //must a rigid body be on one of the objects  **if** (!inAction) // since we are using auto lerping , we must not allow the trigger to always interact with solution  **return**;  **if** (TasksManager.MissionID <**3**)  {  //print("other: " + other.name);  UIObjsHandler.instance.AlertString = " Check Safety Rules and Fire the Bunsen first ";  **return**;  }  **if** (prevAmountOfSolinthisDrpr + ChoosenAmountOfSolute <= **100**)  {  AmountDrpNeeds = ChoosenAmountOfSolute;  }  **else**  {  AmountDrpNeeds = **0f**; //it will absorb from bottle but ( 0 amount ) so no bad effect ( on bottle ),, // the dropper itself have a condition to not fill itself if nearfull }  //( Solu\_Interact ) handles visuals and action in another side like cylinders or soultions bottles  **var** SolutionInteract = other.GetComponentInParent<Interface\_interactbles>();  **if** (SolutionInteract != **null**)  {  **try**  {  //bool done = SolutionInteract.Solu\_Interact(AmountDrpNeeds, prevAmountOfSolinthisDrpr, oldBlendClone.GetComponent<BlendInfo>().materialblendColor, oldBlendClone.GetComponent<BlendInfo>().thisBlendType, out thisDropperIsAbsorbinSol);  **bool** done = SolutionInteract.Solu\_Interact(AmountDrpNeeds, prevAmountOfSolinthisDrpr, oldBlendClone.GetComponent<BlendInfo>(), **out** thisDropperIsAbsorbinSol);  **if** (!done)  **return**;  }  **catch**  {    **bool** done = SolutionInteract.Solu\_Interact(AmountDrpNeeds, prevAmountOfSolinthisDrpr, **new** BlendInfo(), **out** thisDropperIsAbsorbinSol);  **if** (!done)  **return**;  }  }  **else** **return**;    HandlingDropperVisualsAndAction(SolutionInteract.GetBlend()); //must be downhere cuz of guard-clauses  }  GameObject oldBlendClone;  **void** **HandlingDropperVisualsAndAction**(GameObject interactBlend)  {  **if** (thisDropperIsAbsorbinSol)  {  **float** TotalEndInDropper = (prevAmountOfSolinthisDrpr + ChoosenAmountOfSolute);  //print("dropper is absorbing from lotion bottle ");  **if** (TotalEndInDropper <= **100**)  {  BlendInfo oldBlendCloneBInfo;  **if** (!oldBlendClone)  { //first time absorbing  oldBlendClone = GameObject.Instantiate(interactBlend, BlendTPos.position, BlendTPos.rotation);  oldBlendClone.transform.localScale = BlendTPos.localScale \* **2.5f**;  oldBlendClone.transform.SetParent(thisTransform);  oldBlendCloneBInfo = oldBlendClone.GetComponent<BlendInfo>();  }  **else**  {  oldBlendCloneBInfo = oldBlendClone.GetComponent<BlendInfo>();  **if** (prevAmountOfSolinthisDrpr > **0**)  {    //getting the info of blend from giver(bottles) to dropper is by (Hit object of ontrigger) (but from dropper to taker -cylinder with The Interact Method)  //mix color of solution in dropper (taking percentage of each color in calculations)  oldBlendCloneBInfo.material.color = ((oldBlendCloneBInfo.materialblendColor \* (prevAmountOfSolinthisDrpr / TotalEndInDropper)) + (interactBlend.GetComponent<BlendInfo>().materialblendColor \* (ChoosenAmountOfSolute / TotalEndInDropper)));  oldBlendCloneBInfo.materialblendColor = oldBlendCloneBInfo.material.color; // material.color changes the color for real , but materialBC will still be the old one so  //mix types of solution in dropper  oldBlendCloneBInfo.water = (interactBlend.GetComponent<BlendInfo>().water || oldBlendCloneBInfo.water); // mixing :D means any true here or there is stronger  oldBlendCloneBInfo.starch = (interactBlend.GetComponent<BlendInfo>().starch || oldBlendCloneBInfo.starch);  oldBlendCloneBInfo.IKI = (interactBlend.GetComponent<BlendInfo>().IKI || oldBlendCloneBInfo.IKI);  }  **else**  {  oldBlendCloneBInfo.material.color = interactBlend.GetComponent<BlendInfo>().materialblendColor;  oldBlendCloneBInfo.materialblendColor = oldBlendCloneBInfo.material.color;  oldBlendCloneBInfo.water = interactBlend.GetComponent<BlendInfo>().water ;  oldBlendCloneBInfo.starch = interactBlend.GetComponent<BlendInfo>().starch ;  oldBlendCloneBInfo.IKI = interactBlend.GetComponent<BlendInfo>().IKI;  }  **if** (!oldBlendClone.activeInHierarchy)  oldBlendClone.SetActive(**true**);  oldBlendClone.transform.SetParent(thisTransform);  }  thisDropperBlClone\_skMeshRderer = oldBlendCloneBInfo.skMsRndr;  **if** (thisDropperBlClone\_skMeshRderer)  thisDropperBlClone\_skMeshRderer.SetBlendShapeWeight(**9**, TotalEndInDropper);  prevAmountOfSolinthisDrpr += ChoosenAmountOfSolute; //cus done now change prev  }  isEmpty = **false**;  }  **else**  {  **if** (ChoosenAmountOfSolute > **3**)  {  UIObjsHandler.instance.AlertString = "Wrong , 3 ml only required , Set in Wanted Amount Panel";  }  thisDropperBlClone\_skMeshRderer.SetBlendShapeWeight(**9**, **0**);  prevAmountOfSolinthisDrpr = **0**;//reset is a must  **if** (oldBlendClone.activeInHierarchy)  oldBlendClone.SetActive(**false**);  }  }  }  } |

# CHAPTER 4

# RESULTS AND DISCUSSION

## 4.1 Results Analysis and Discussion

## In this section we will see how our chemical experiment work using

## Unity Game Engine

## First choose language from Drop down list and hit start screen

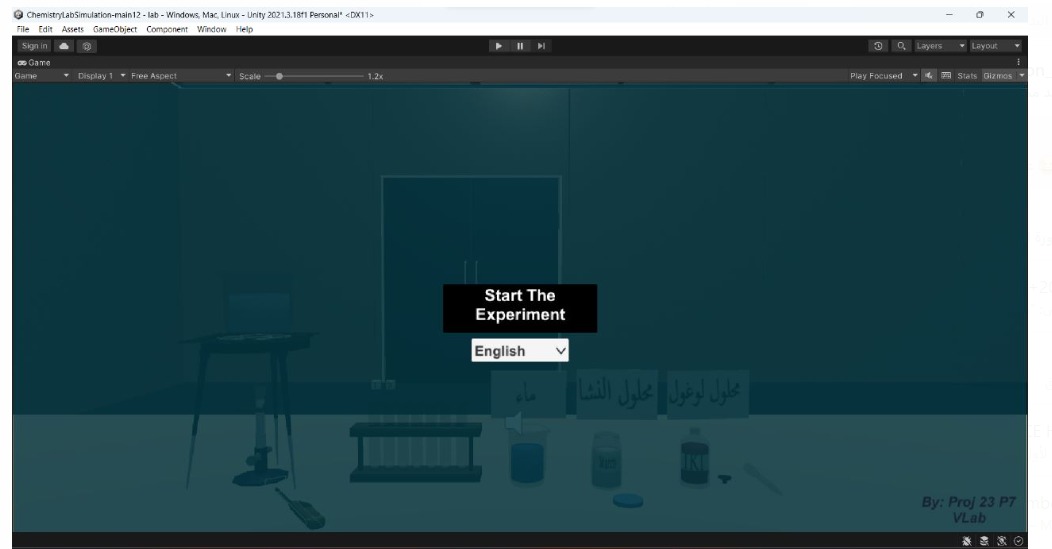


Figure 4.1 First UI Screen

* There is automatic audio will play, instruction panel to guide user

In each step, Amount screen , time screen and wrong screen.

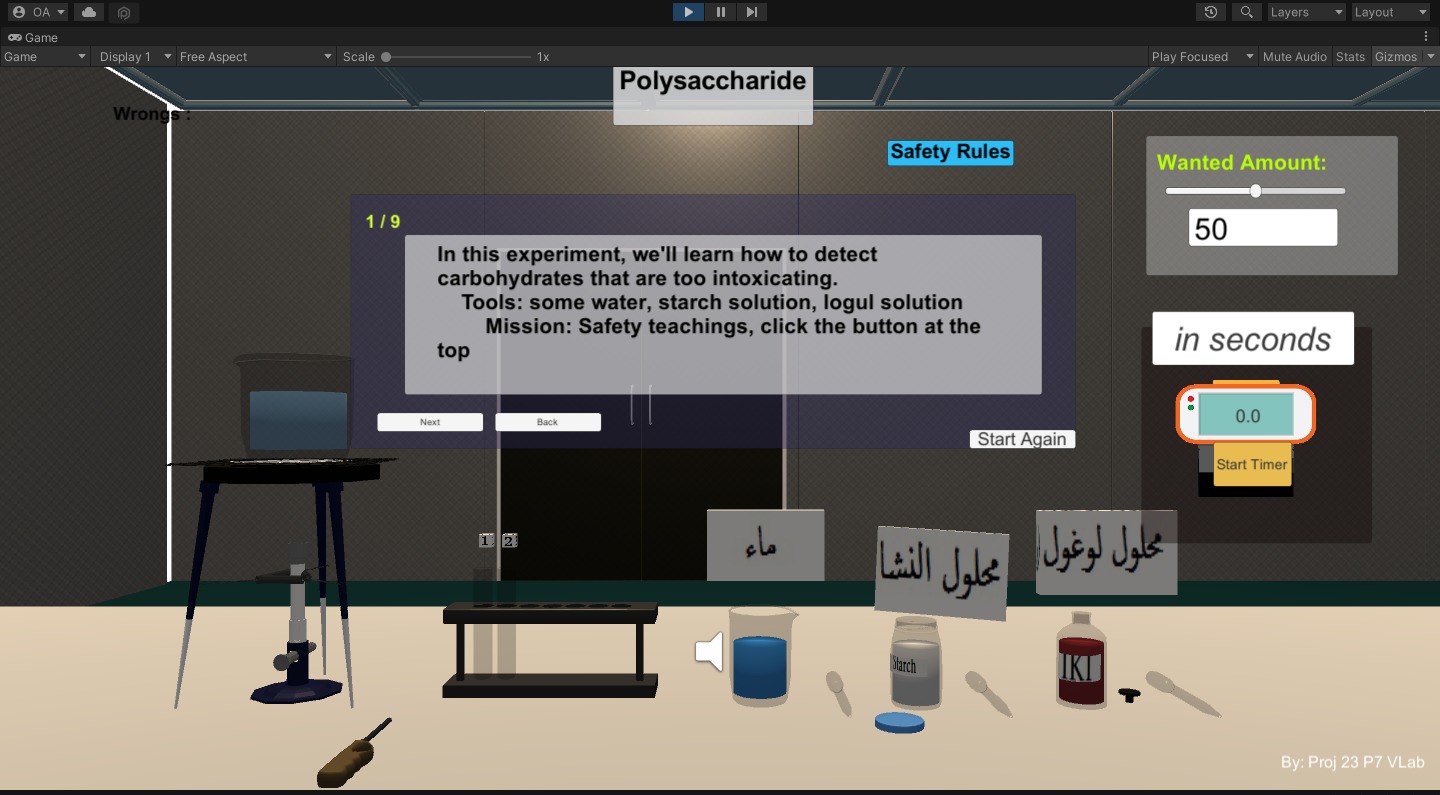


Figure 4.2 First Instruction

* Click on safety Rules button to go to safety rules and click on same

Button to return back to do experiment.



Figure 4.3 Safety Rules In Game Mode

* Bunsen Fire



Figure 4.4 Second Instruction

* Add 3 ml of water and starch to tube1 and tube2 respectively

Set amount using amount screen.



Figure 4.5 Third Instruction

* Add iodine solution to both tubes by same way



Figure 4.6 Fourth Instruction

* Put tube in hot bath and set timer using timer screen



Figure 4.7 Fifth Instruction

* Finally we tack tubes and notice that tube with starch in it , its color changed



Figure 4.8 Sixth Instruction

## 4.2Testing Data

## In this section we will try to interact with our experiment with

## Wrong sequence of actions test .

## hit start screen and put water in tub without click on safety rules.

## 



Figure 4.9 Test 1

* if safety rules comes first but second put water in tub without Fire

Bunsen fire



Figure 4.10 Test 2

* if we try to put starch in tube instead of water



Figure 4.11 Test 3

* if we try to put water in tube instead of starch



Figure 4.12 Test 4

* try any amount not 3ml

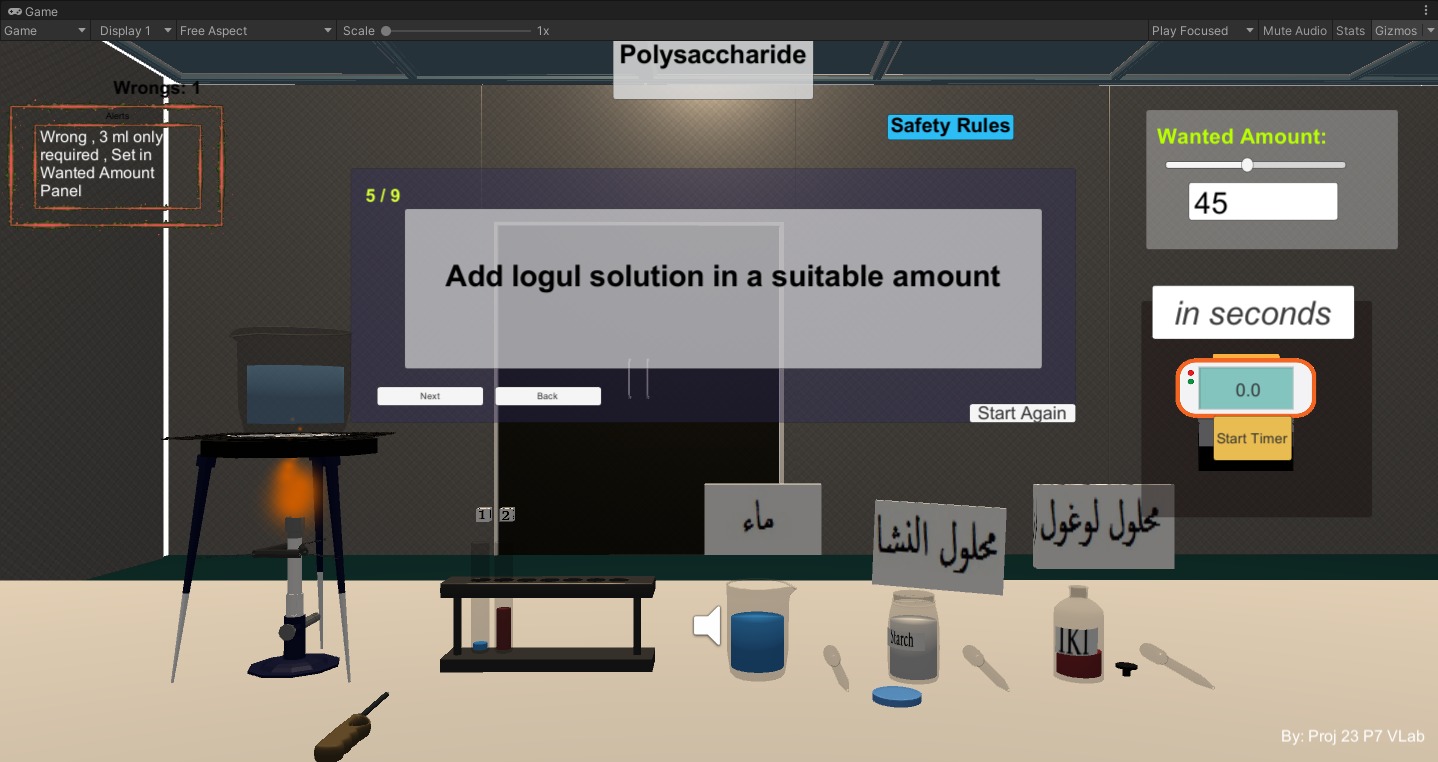


Figure 4.13 Test 5

* Try any time not (120 sec)

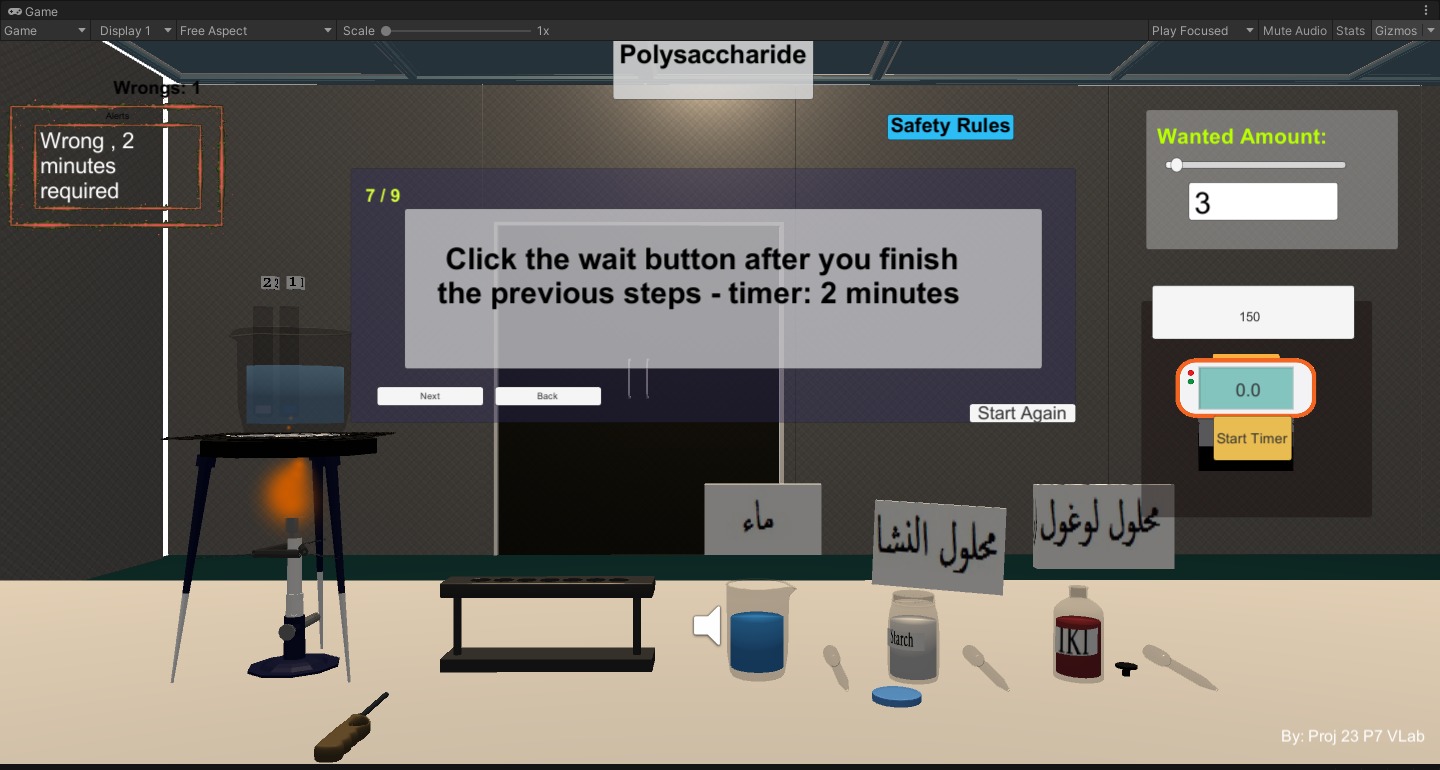


Figure 4.14 Test 6

# CHAPTER 5

# CONCLUSION AND FUTURE WORK

## Conclusion

## designed a chemical experiment using the Unity game engine this experiment

## is interactive little bit interactive with users to make it easier .

## Future Work

## Add many experiment

## Add many tools

## Make teacher able to create any experiment he want

## Improve user interface

## Add video tutorial to each experiment

## Rung project on different platforms

## 

# 

# REFERENCES

1. Developing Virtual Reality Applications: Foundations of Effective Design. Alan B. Craig, William R. Sherman , Jeffrey D. Will (2009).
2. Augmented Reality: Where We Will All Live. Jon Peddie (2017).
3. Chemistry 2e Book . openstax college , Rice University .

Link: <https://openstax.org/books/chemistry-2e/pages/1-introduction>