Acknowledgement

It has been a great opportunity to gain lots of experience in real time projects, followed by the knowledge of how to actually design and analyze real projects. For that we want to thank all the people who made it possible for students like us. Special thanks to **The Higher institute of computer Science & information systems** for the efforts they did to provide us with all useful information and making the path clear for the students to implement all the education periods in real-time project design and analysis. Furthermore, we like to express our deepest gratitude to our graduation project supervisor
Prof. Dr.AbdEl Majeed Amen Ali for his patience and guidance along the semester. In addition, we would like to express our sincere appreciations to our graduation project second supervisor **T.A. Omnia Khalaf** for his guidance, continuous encouragement and support during the course. Moreover, it is our duty to thank all the testing committee members for their generous discussions and encouragement.

Abstract

Education is defined as the conscious attempt to promote learning in others. Traditionally, analysis of this attempt has centered on direct teaching on the part of teachers. In what constitutes a paradigm shift; however, people now note that learning can be promoted in ways that go beyond direct instruction by a teacher-education now centers around creating a viable, productive learning environment, regardless of how teacher-centric that environment might be. When the term education is combined with entertainment, the term edutainment is coined. Edutainment also called "e-learning" are new methods and practices that enabled learning in faster, more efficient and more entertaining ways. The idea is usually to combine games with learning, using software or interactive courses. A stage of edutainment is to combine Visuals with Auditory. In this context, visuals means not only pictures but also 3D and 2D videos. Multimedia representation of the classroom lessons was available to students. This mode of teaching through Computers is known as Computer Based Training abbreviated as CBT. Through CBT a student will come to know how a system will look alike and how it works. Virtual Labs have considerable educational CBT method because they provide an opportunity to ’learn by doing’. They also provide access to systems which are otherwise inaccessible for reasons such as safety, cost and size. Users can explore a variety of what if scenarios by changing the input and observing the effect on the output. Thus, the virtual labs have lots of advantages.

1

INTRODUCTION

1. The importance of the computer and its uses

Not long ago, our daily business and our practice of our life and its requirements such as education, learning, communication with government agencies, companies and other various means of life required time and effort. In our daily life, almost nothing in our daily work is devoid of using the computer, whether inside or outside the home. We find it in the field of education, learning, education, detention, communication, or in the markets and others in all areas of life. With the tremendous advances in computer technology and information technology, it has become more suited to the many demands imposed by human industry and development, and most of them.

And that the computer has systems for communicating with the Internet that made it easy to obtain information, saved time and effort, and was easy to communicate with others.

Computer use became necessary in our lives. And what we are witnessing of the tremendous and rapid development of computer technology calls us to activate it in the field of education in an innovative way, as it is no longer a field of knowledge except that the computer has an important role in it. As the computer is the backbone of the developmental, economic, social and educational process. It is the main tool in the fast data processing.

2. Education VS Coronavirus

The world is currently witnessing a momentous event that may threaten education with a huge crisis, perhaps the most dangerous in our contemporary time. As of March 28, 2020, the Coronavirus (Covid-19) pandemic has caused more than 1.6 billion children and youth to drop out of education in 161 countries, that is, nearly 80% of students enrolled in school globally. This came at a time when we are already suffering from a global educational crisis, as there are many students in schools, but they do not receive the basic skills they need in working life. The World Bank's index of "learning poverty" - or the percentage of students who cannot read or understand at the age of 10 - shows that the proportion of these children in low and middle-income countries prior to the outbreak of the virus was 53%. If we do not act, this pandemic may worsen that outcome.

But what are the direct effects on children and youth that cause us concern at this stage of the crisis? 1. Learning losses. 2. Increase dropout rates. 3. Children do not get the most important meal of the day. What is more, the inequality in educational systems that most countries suffer from, and there is no doubt that these negative effects will affect poor children more than others. As if calamities do not come to them individually.

3. Product Scope

The "Virtual Lab" is a web-based application which helps students to learn and perform their school-scientific experiments as if they actual in physical labs.

With VL, students can watch the explanations for the experiments they want, perform experiment anytime and anywhere (i.e., at school with teacher or at home by themselves) via step-by-step education method. And teacher can do any experiment with his students.

2 Planning and Analysis

1. How did we find a program to work the virtual lab?

And after we decided to make the virtual lab, we started searching for programs and tools that help us make a 3D virtual lab, and the student can easily do his experiments and can dispense with the lab itself.

Therefore, after a long search and the help of **Prof. Dr. /: Abdel Majeed Amen Ali** us, we found that the Unity 3D program helps us completely and greatly to create any experience in the virtual lab, where there are many advantages of the Unity 3D program through which you can make 3D models where the student feels as if he is in the lab and also provides us with writing codes Which makes these models move and interact with each other and the language used in Unity is C#.

2. Our Plan is:

1. We started to learn how to design the tools used in doing chemical experiments using the Unity program. Indeed, we have designed the experimental tools we use in the virtual lab.

2. Then after we finished working with all the tools used, we started learning to write code using C# language.

3. Then we started linking script files and the Objects used in the virtual lab.

4. Choose a curriculum, so choose the sixth grade of primary school to do experiments. An example of this is two experiments, the first is the preparation of carbon dioxide gas and the second is the preparation of oxygen gas.

5. Then we began distributing tasks to the members of the team, and we began to divide the experiment into two parts, code and design the tools used in conducting the experiment, and also the other part of the team to test the experiment if something was found to modify.

3. Functional Requirements

Our web site can used by student or teacher:

Teacher

- Can view our web site and see all experiment.
- Choose any experiment to do it with his student.
- Can enter to experiment and do this experiment with his students step by step.
- See in final result of experiment.

Student

- Can view our web site and see all experiment.
- Choose any experiment to do.

- May do this experiment with his teacher or with himself easy as before experiment start, there is sound explain how to do this experiment.
- Can enter to experiment and do this experiment step by step.
- See in final result of experiment.

4. Non-Functional Requirements

Performance Requirements

- The system should be having quickly response to user.
- The software should have the capacity for doing any number of operations or entering a huge data.

Availability

- The system shall be made available to the user/administrator year round.

Usability

- The software should have graphical user interface.
- The system must be responsive to other devices that it run on.

Reliability

- Reliability is the ability of the system to perform its required functions under start conditions for specific period.
- The software should be available for any requests when the users use the system.
- The system should not failure when doing operations on the system.

Maintenance

- The system should be able to maintain from time to time to be up-to-date and repair errors if found in the system.

5. Need for the new system

1. X Lab

This is a mobile application that works using the VR glasses in order to provide all the scientific experiments for students. Here the problem is that it is very expensive due to the use of the glasses. As for Virtual lab, you do not need the glasses because all the experiments are presented in 3D, so they do not need glasses.

2. Praxilabs

   PraxiLabs was built, and is continuously developed and improved, by a dedicated team of programmers and education specialists who understand the importance experimentation in science education. We've developed a product that makes virtual science labs accessible, usable, and affordable for educational institutions and schools. PraxiLabs not only provides an immersive virtual lab experience, but adds enriched content that provides students more understanding and knowledge. The PraxiLabs team does not simply offer schools, colleges, and institutions this fantastic solution, it partners with them to ensure those organizations are utilizing it for the best of its capabilities. Here is the problem in PraxiLabs. Because they direct these experiences to students with a master's or doctorate or in universities and not to

students at Preparatory stage, and this is not appropriate for them, so we made Virtual
Lab in order to help them to do their experiments easily and conveniently.

3. Labster

   Labster is a company dedicated to developing fully interactive advanced lab simulations based on mathematical algorithms that support open-ended investigations. We combine these with gamification elements such as an immersive 3D universe, storytelling and a scoring system which stimulates students' natural curiosity and highlights the connection between science and the real world. The labs are being used by California State University, Harvard, Gwinnett Technical College, MIT, Exeter University, University of New Haven, Stanford, University of New England, Trinity College, University of Hong Kong and Berkeley among others internationally. The problem here is that it is not suitable for middle school students.

4. ChemCollective

   As a project in the National Science Digital Library (NSDL), the ChemCollective's goals are to support a community of instructors interested in improving chemistry education through interactive and engaging online activities. Paper-and pencil homework typically emphasizes applying formulas a process that can become routine and disconnected from the reality and fun of doing chemistry. In contrast, simulation-based exercises offer new ways to promote learning and motivation. Interactive exercises can allow students to explore and reinforce fundamental concepts in contexts that are increasingly complex, realistic, and engaging. Our goal is to create flexible, interactive learning environments where college and high school students can approach chemistry more like practicing scientists. The problem here is that it is not suitable for middle school students and requires more programs to download to the device in order for the student to do the experiment.

5. Virtual Labs

   Virtual Labs will provide to the students the result of an experiment by one of the following methods (or possibly a combination). Modeling the physical phenomenon by a set of equations and carrying out simulations to yield the result of the particular experiment. This can, at-the-best, provide an approximate version of the 'real-world' experiment. Providing measured data for virtual lab experiments corresponding to the data previously obtained by measurements on an actual system. Remotely triggering an experiment in an actual lab and providing the student the result of the experiment through the computer interface. This would entail carrying out the actual lab experiment remotely. The problem here is that it is not suitable for middle school students.

6. Advantages of the new system

Advantages of the virtual laboratory it helps students to teach the experiment with a simplified explanation through the sound and steps to how to do it.

7. Feature map

The use case we guarantee for each of our users

- Teachers
- Students

Views or Pages

- Home Page
- Every page for every experiment

First student or teacher enter to Virtual Lab then can choose needed experiment to do it. Second student or teacher can do experiment with steps and sound in the first to how to do this?

8. Technical requirements and specification

Backend

- C#

Tools

- Using unity 3d environment

Why Unity 3d?

Using this program allows us to design things in 3D and also allows us to write codes using C# to make these things move like in chemical experiments. We designed tools using this program and then write code on it to make these materials interact with each other.

Frontend

- HTML
- CSS

9. Diagrams

Context Diagram

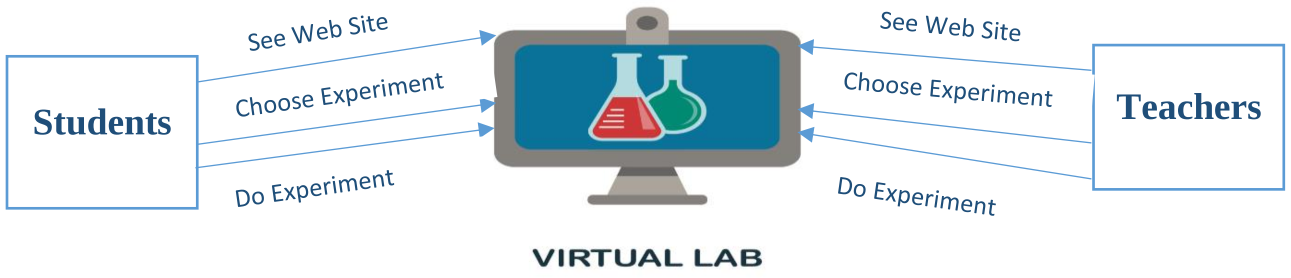


Figure 2.1: Context Diagram

A context diagram defines the boundary between the system, or part of a system, and its environment, showing the entities that interact with it

Data Flow Diagram (DFD)

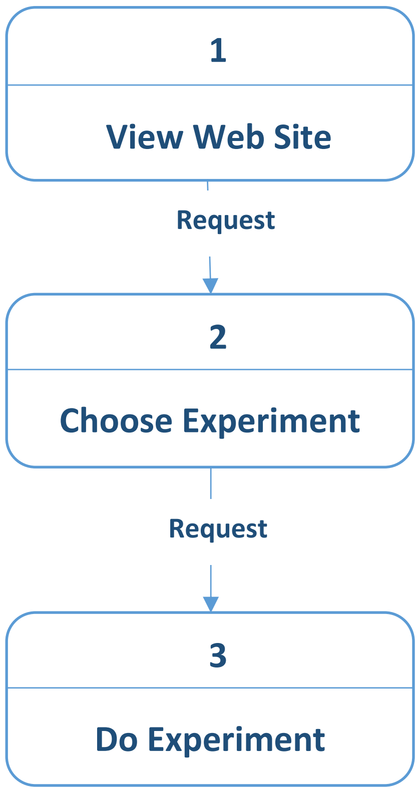


Figure 2.2: Data Flow Diagram

A data-flow diagram is a way of representing a flow of data through a process or a system. The DFD also provides information about the outputs and inputs of each entity and the process itself

Activity Diagram

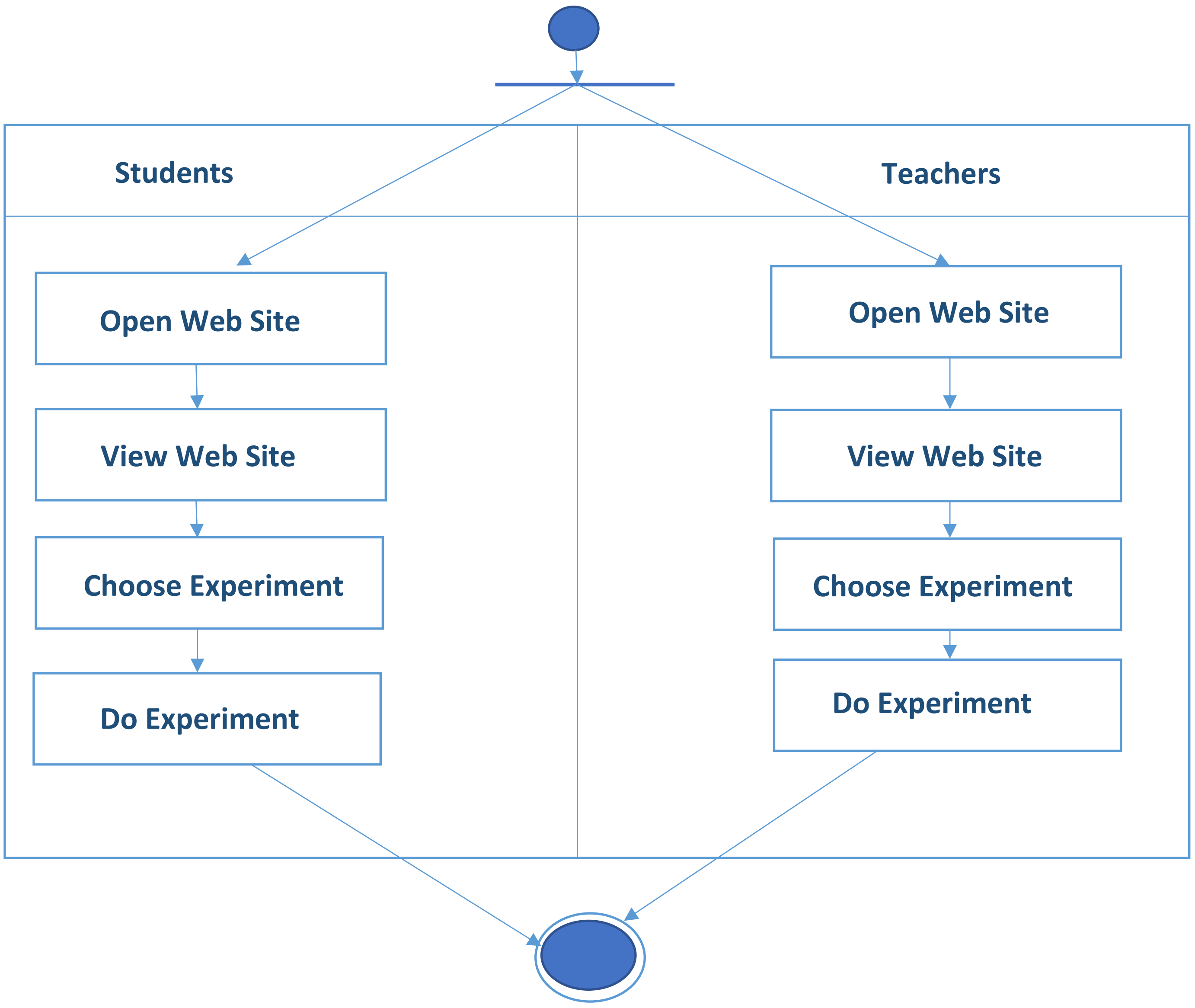


Figure 2.3: Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency.

Use Case Diagram

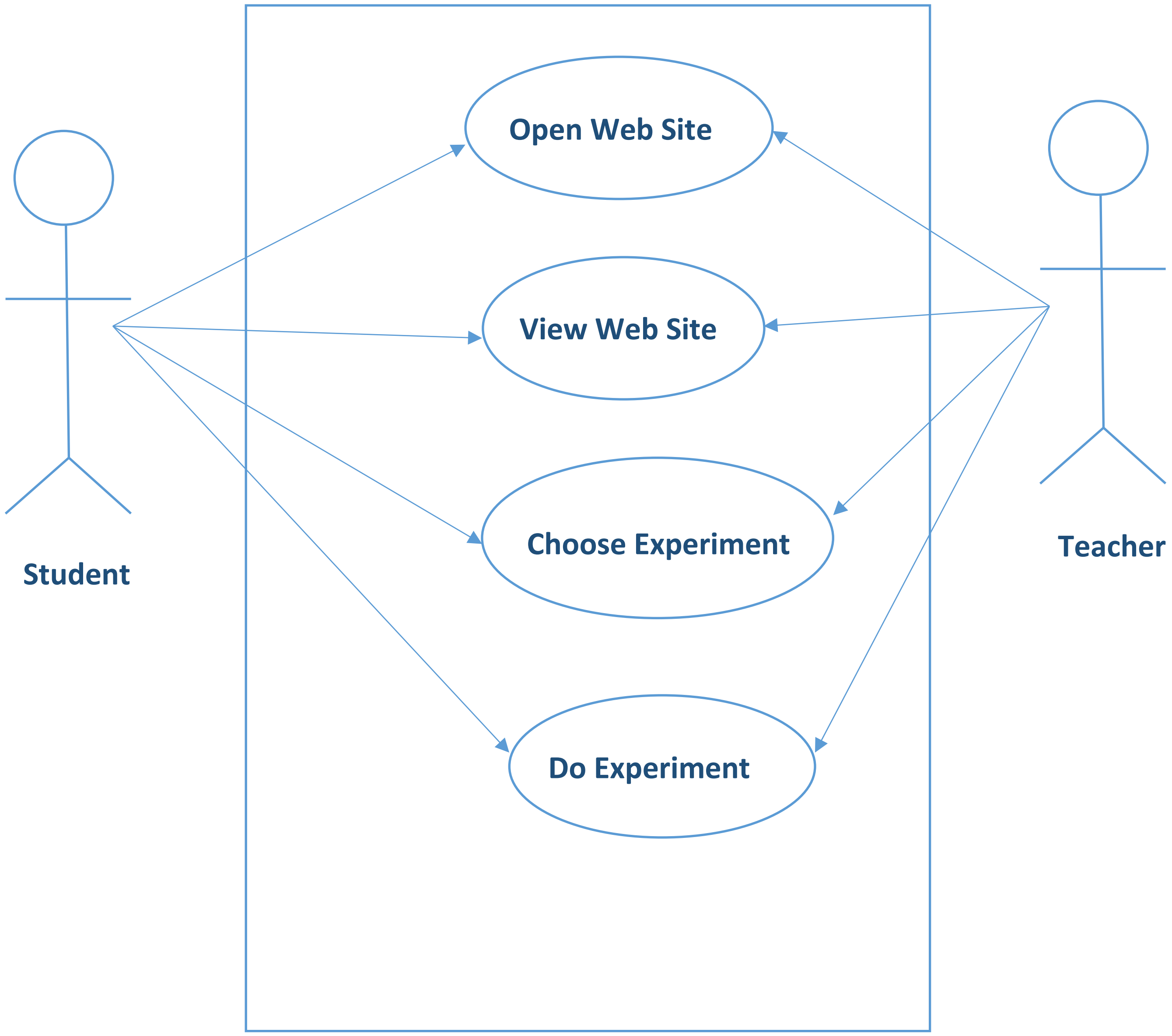
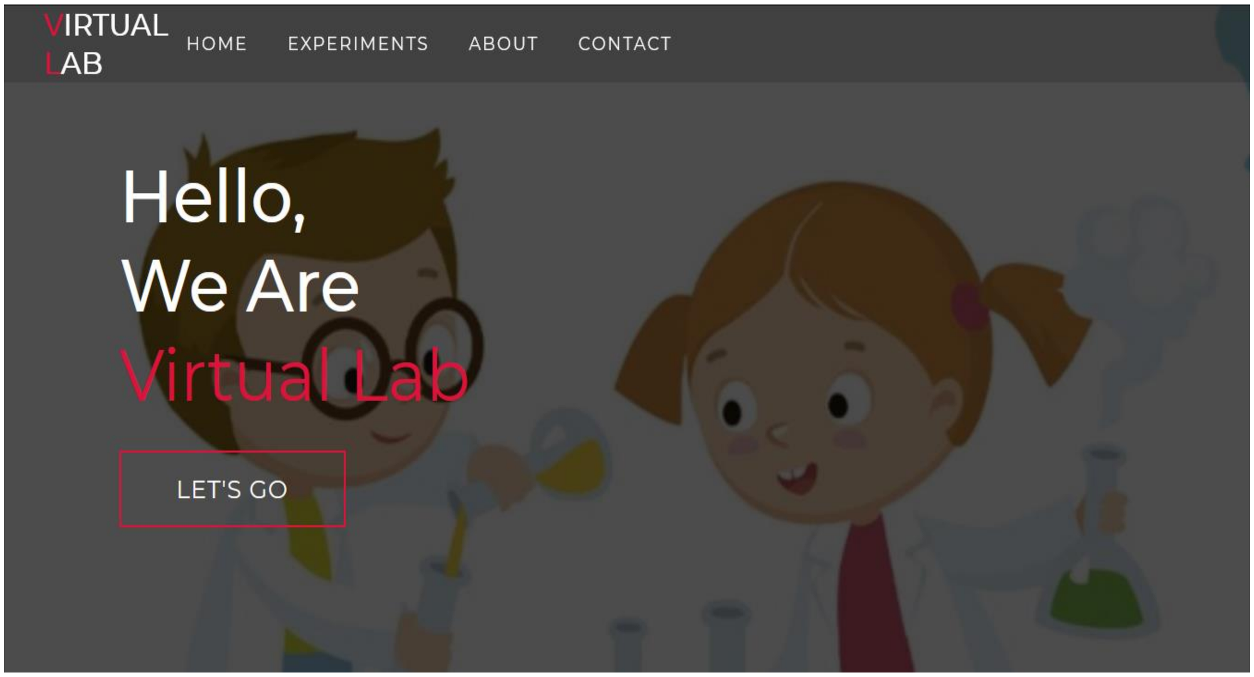


Figure 2.4: Use Case Diagram

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has.

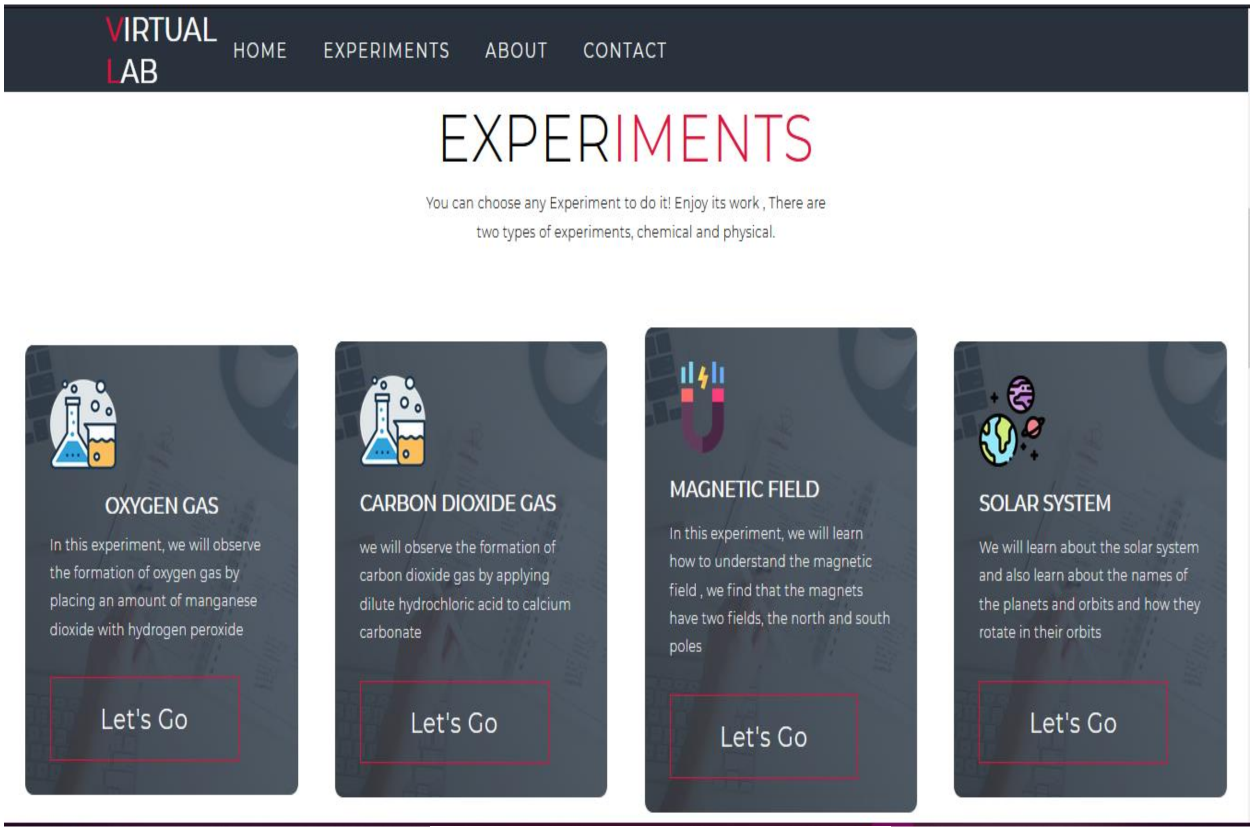
3 Implementation

User Interface



Hello,
We Are
Virtual Lab

Figure 3.1: Home Page



EXPERIMENTS

You can choose any Experiment to do it! Enjoy its work , There are two types of experiments, chemical and physical.

OXYGEN GAS

In this experiment, we will observe the formation of oxygen gas by placing an amount of manganese dioxide with hydrogen peroxide

CARBON DIOXIDE GAS

we will observe the formation of carbon dioxide gas by applying dilute hydrochloric acid to calcium carbonate

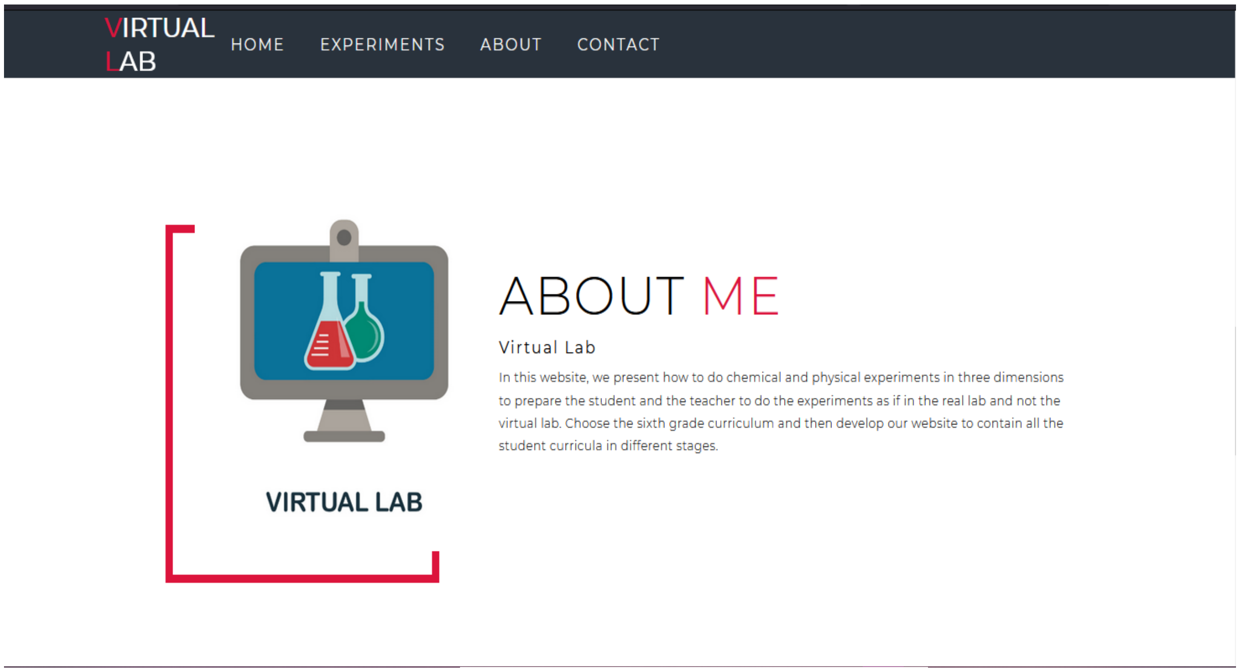
MAGNETIC FIELD

In this experiment, we will learn how to understand the magnetic field , we find that the magnets have two fields, the north and south poles

SOLAR SYSTEM

We will learn about the solar system and also learn about the names of the planets and orbits and how they rotate in their orbits

Figure 3.2: Experiments

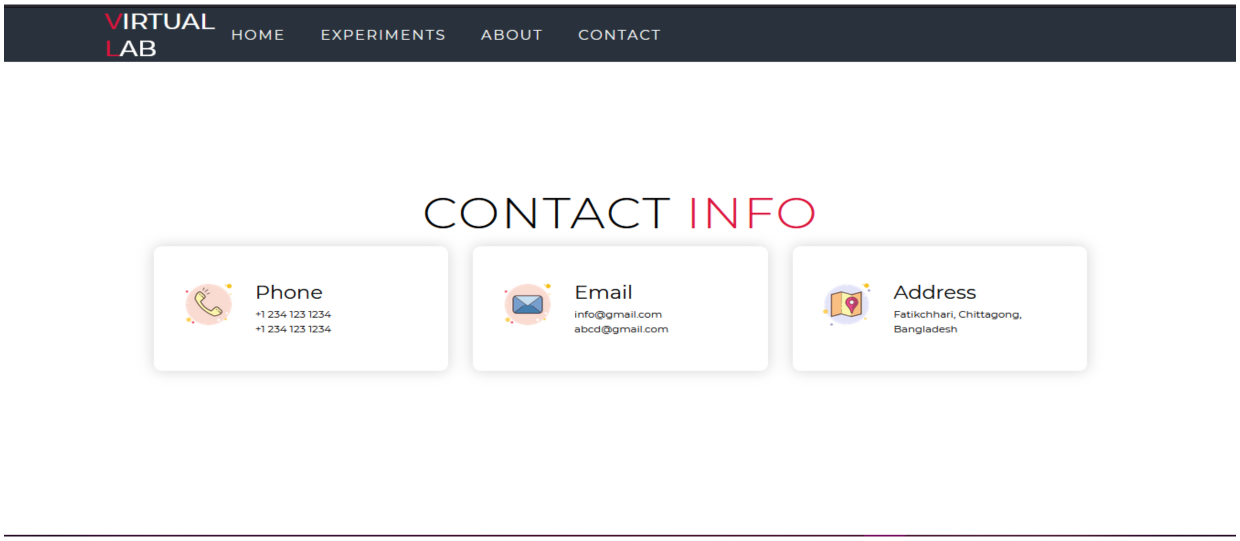


ABOUT ME

Virtual Lab

In this website, we present how to do chemical and physical experiments in three dimensions to prepare the student and the teacher to do the experiments as if in the real lab and not the virtual lab. Choose the sixth grade curriculum and then develop our website to contain all the student curricula in different stages.

Figure 3.3: About Me



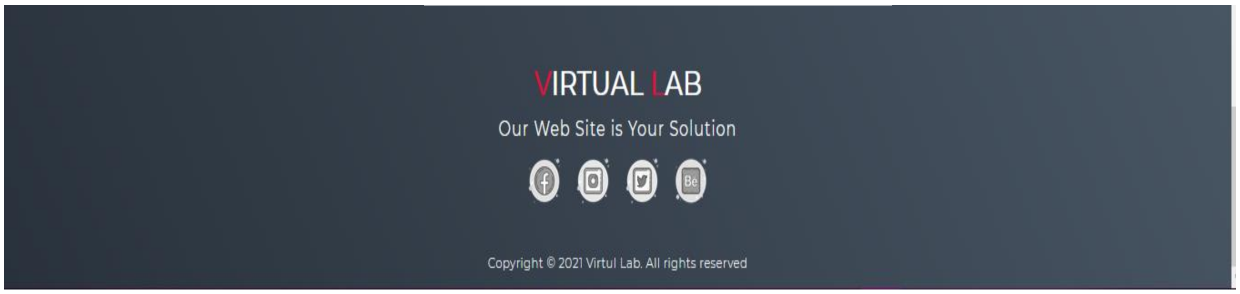
CONTACT INFO

Phone
+1 234 123 1234
+1 234 123 1234

Email
info@gmail.com
abcd@gmail.com

Address
Fatikchhari, Chittagong, Bangladesh

Figure 3.4: Contact Information



Our Web Site is Your Solution

Copyright © 2021 Virtul Lab. All rights reserved

Figure 3.5: Footer

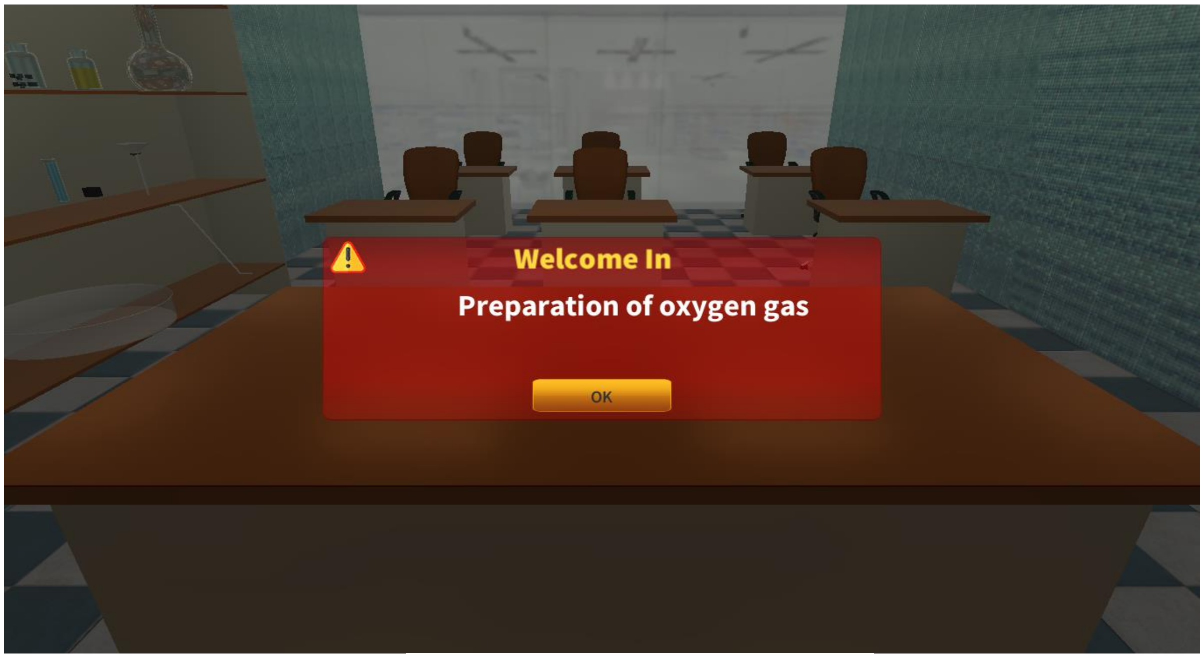


Figure 3.6: Experiment Page

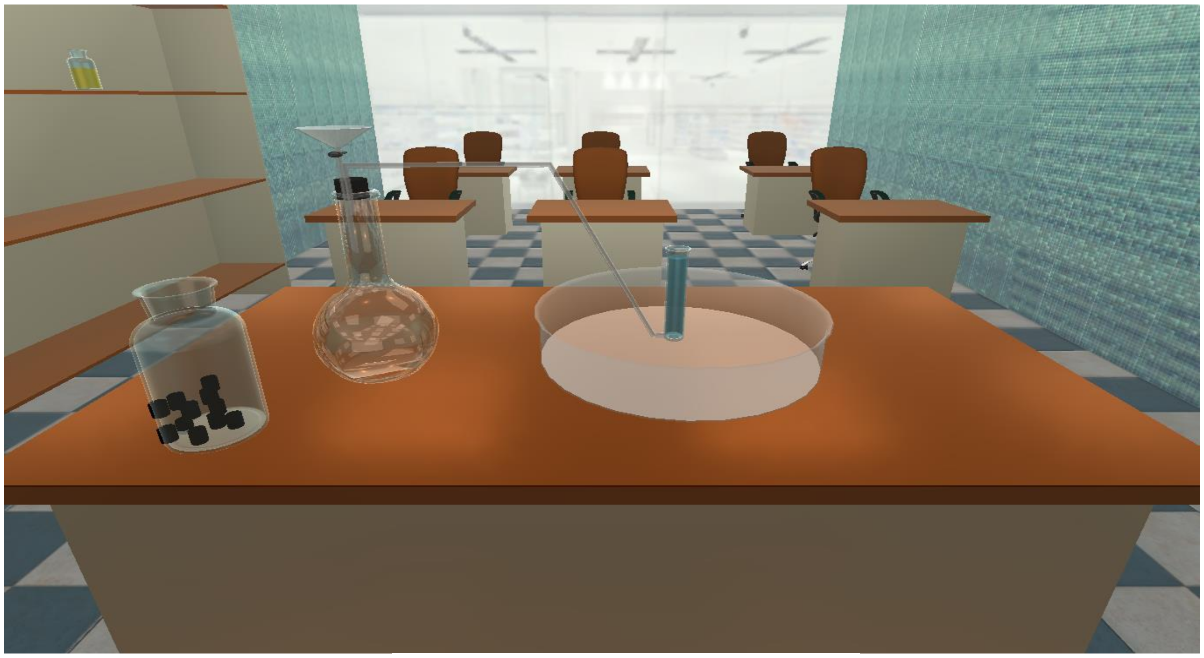


Figure 3.7: Tools of Experiment

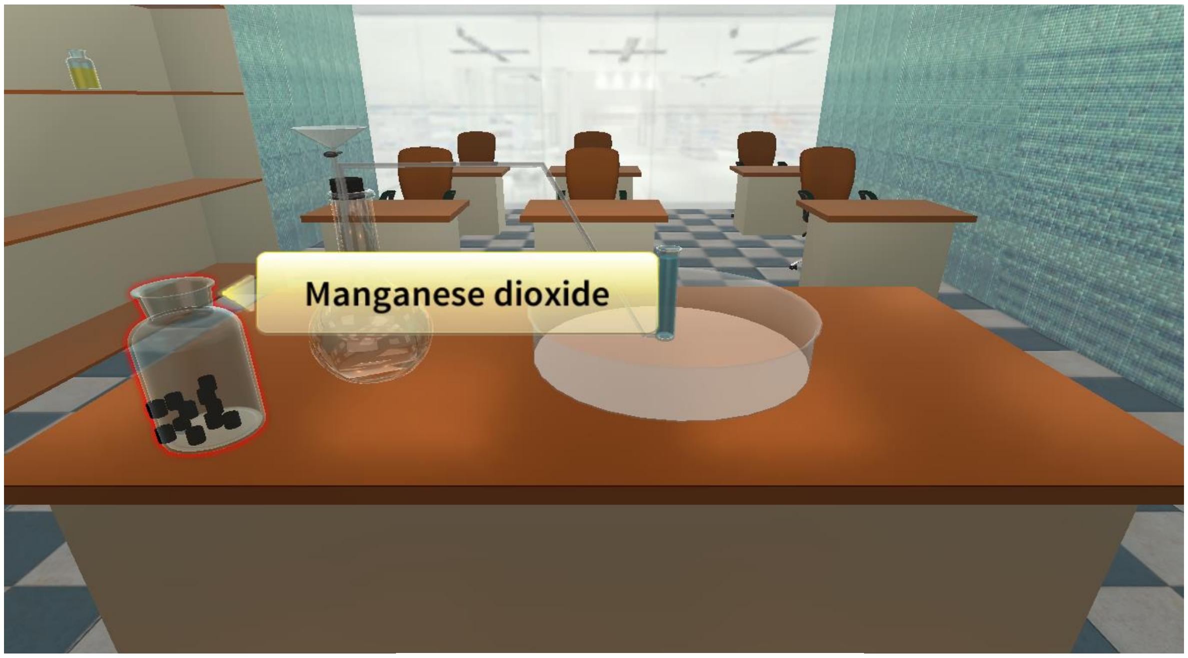


Figure 3.8: How to do Experiment

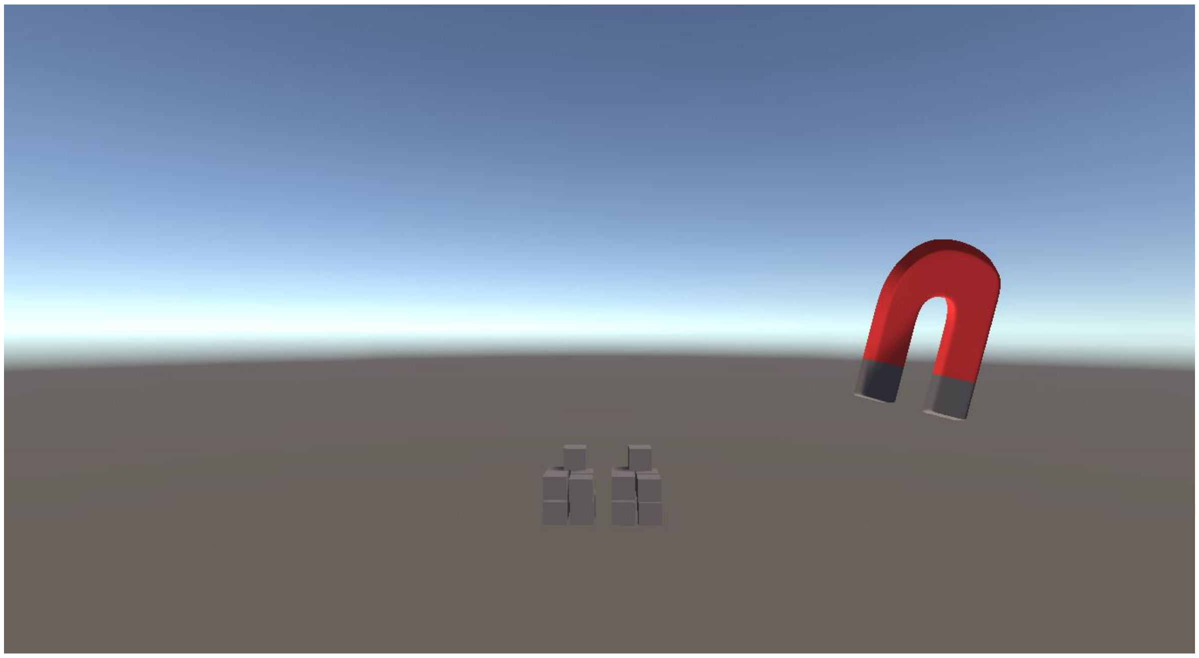


Figure 3.9: Magnet Experiment

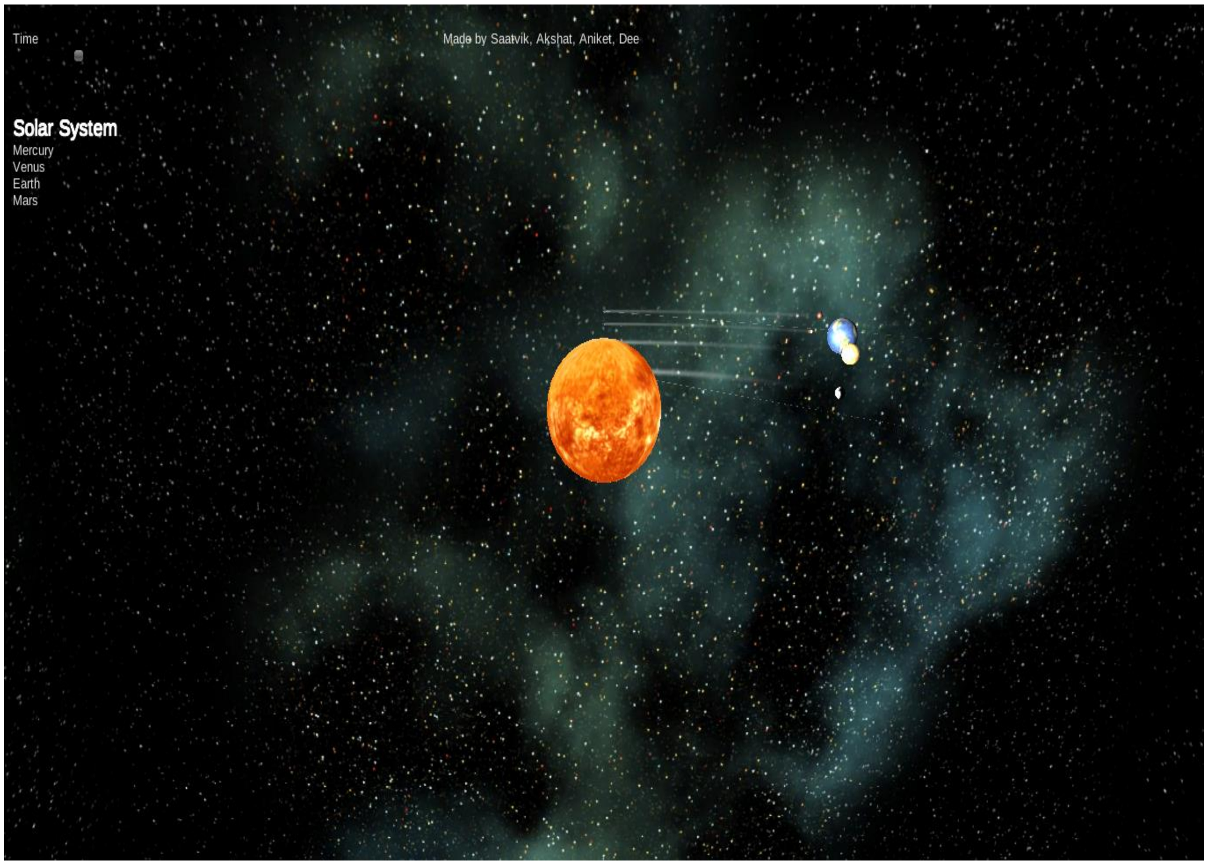


Figure 3.10: Solar System

4 Conclusion

Through our website, the student or teacher can perform the experiment with ease without any risks to the student in the lab without the teacher's attention. And also on our site, we designed the experiment in three dimensions so that the student prepares as if he is in the lab and has no effect on not understanding the experiment.

And each experiment has a sound recording to explain to the student how to do the experiment without errors and also steps for doing this experiment. First, the student enters our website and chooses the experiment to be performed, and then does it with ease.

Future Work

- Account for Each Student and Teacher.
- Chat between Students and Teachers.
- Video for Each Experiment.
- Exams for Students.
- Education Games.

References

[1] Paul Ammann and Jeff Offutt. Introduction to software testing. Cambridge University Press, 2016.

[2] Chem. ChemCollective. URL http://www.chemcollective.org/.

[3] Dave McComb. The Data-Centric Revolution: Data-Centric vs. Data-Driven – TDAN.com. URL https://tdan.com/ the-data-centric-revolution-data-centric-vs-data-driven/20288.

[4] Labster. Labster | 100+ virtual labs for universities and high schools, 2011. URL https://www.labster.com/.

[5] Glenford J Myers, Corey Sandler, and Tom Badgett. The art of software testing. John Wiley & Sons, 2011.

[6] M Packer. Virtual Labs Light. URL http://www.vlab.co.in/www.edmark.com.

[7] Chintakayala Padmini. Beginners guide to software testing. Indianapolis: Lulu Enterprises, Inc, 2006.

[8] PAUL BARNES. Software Costs Estimation Tutorial | Toptal. URL https://www.toptal.com/agile/ software-costs-estimation-in-agile-project-management.

[9] Praxilabs. Virtual Lab | Praxilabs 3D Simulations of Science | Praxilabs. URL https://praxilabs.com/en/https://www.praxilabs.com/en/.

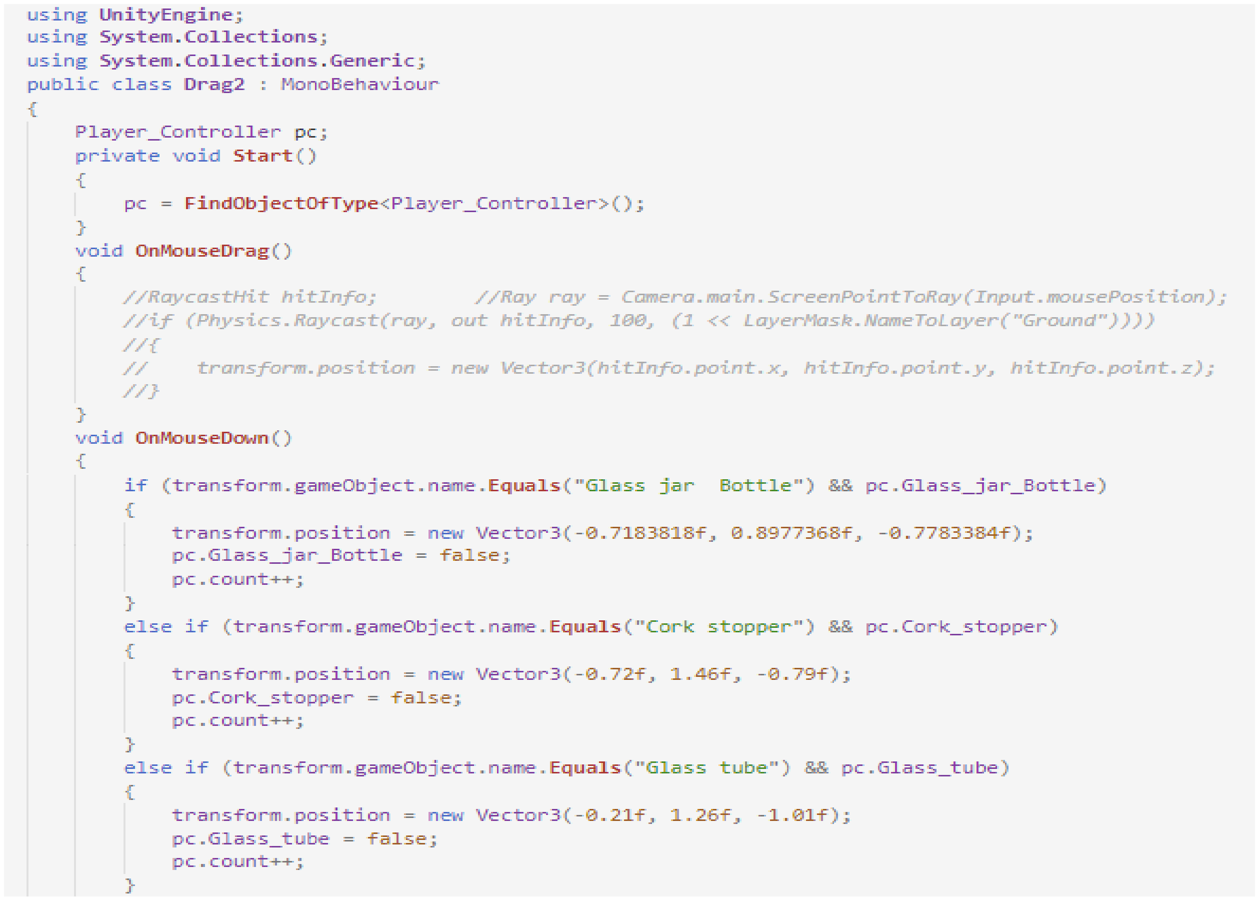
[10] Srs. SRS Documents: Requirements and Diagrammatic Notations | Coursera. URL https://www.coursera.org/learn/srs-documents-requirements.

[11] Xlab. Xlabs - Home. URL https://m.facebook.com/Xlabs.eg/.

[12] Zapier Team. The Ultimate Guide to Project. Technical report, 2019. URL https://www.projectmanager.com/project-planning.

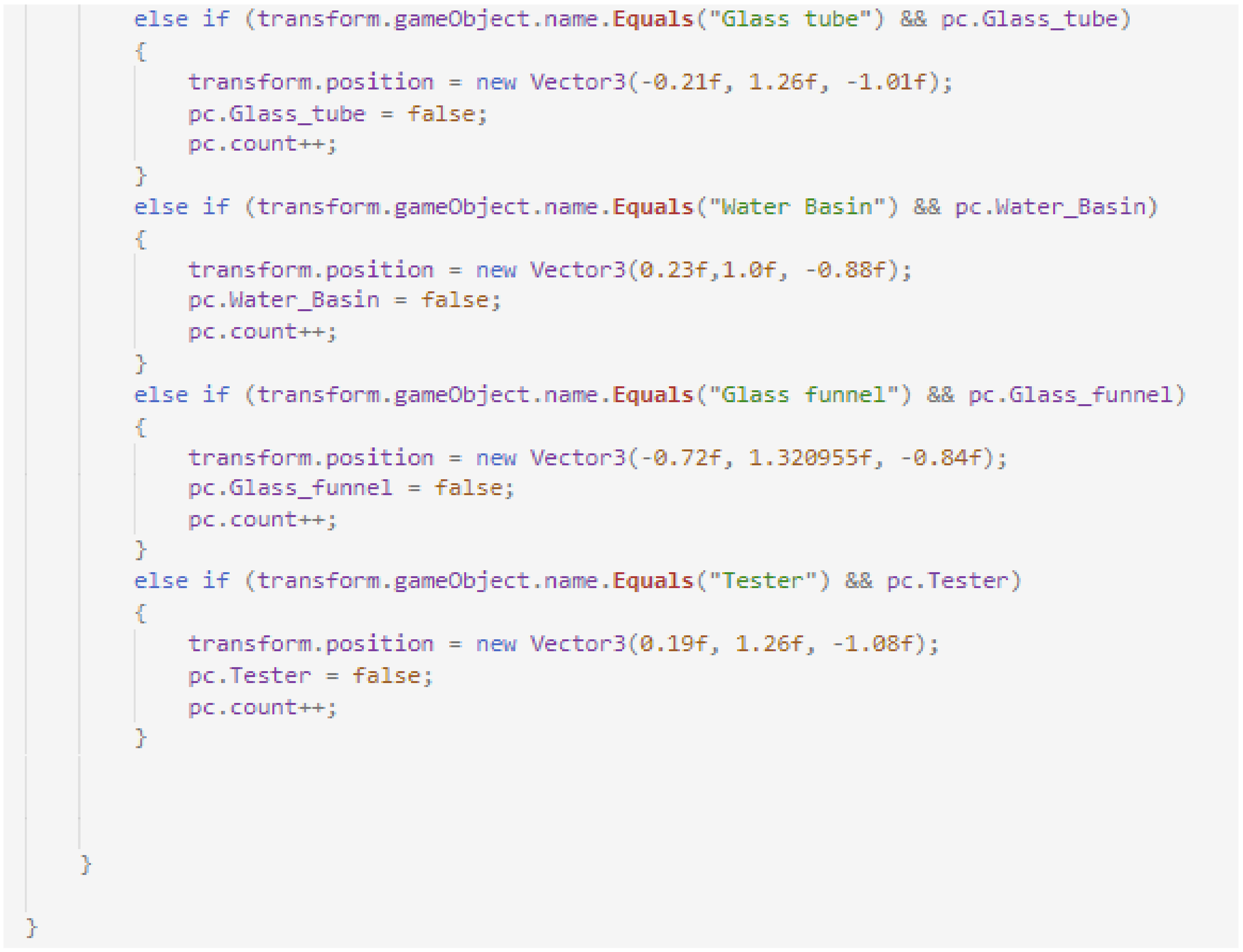
Appendix

Backend



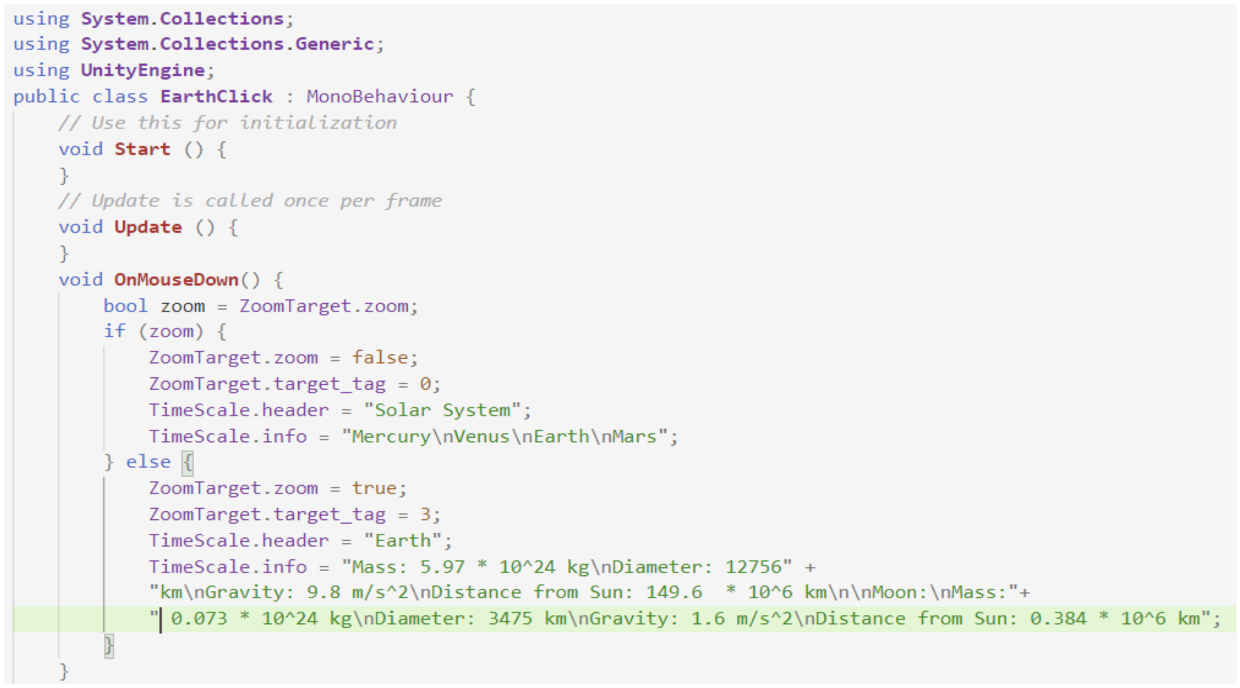
```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
public class Drag2 : MonoBehaviour
{
    Player_Controller pc;
    private void Start()
    {
        pc = FindObjectOfType<Player_Controller>();
    }
    void OnMouseDrag()
    {
        //RaycastHit hitInfo;         //Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
        //if (Physics.Raycast(ray, out hitInfo, 100, (1 << LayerMask.NameToLayer("Ground"))))
        //{
        //    transform.position = new Vector3(hitInfo.point.x, hitInfo.point.y, hitInfo.point.z);
        //}
    }
    void OnMouseDown()
    {
        if (transform.gameObject.name.Equals("Glass jar  Bottle") && pc.Glass_jar_Bottle)
        {
            transform.position = new Vector3(-0.7183818f, 0.8977368f, -0.7783384f);
            pc.Glass_jar_Bottle = false;
            pc.count++;
        }
        else if (transform.gameObject.name.Equals("Cork stopper") && pc.Cork_stopper)
        {
            transform.position = new Vector3(-0.72f, 1.46f, -0.79f);
            pc.Cork_stopper = false;
            pc.count++;
        }
        else if (transform.gameObject.name.Equals("Glass tube") && pc.Glass_tube)
        {
            transform.position = new Vector3(-0.21f, 1.26f, -1.01f);
            pc.Glass_tube = false;
            pc.count++;
        }
```

Figure 4.1: How to Drag Objects part 1



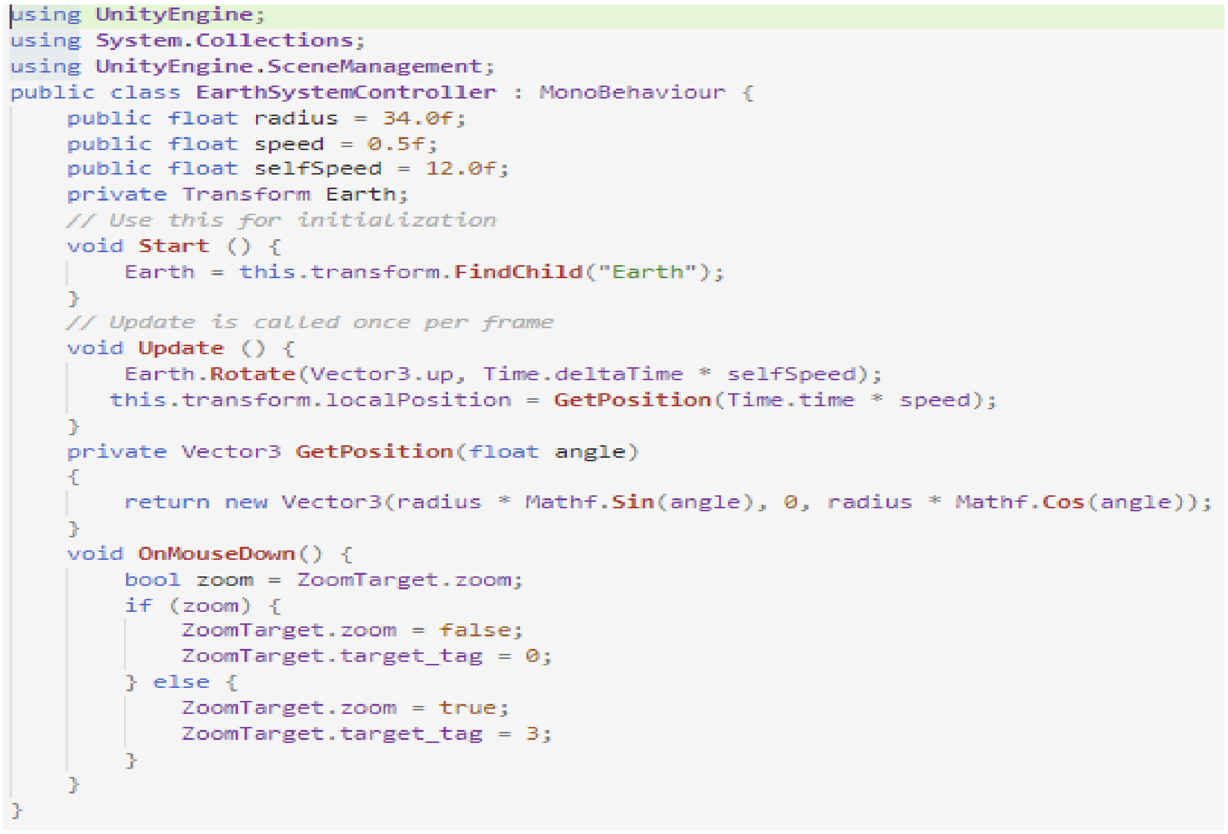
```
            else if (transform.gameObject.name.Equals("Glass tube") && pc.Glass_tube)
            {
                transform.position = new Vector3(-0.21f, 1.26f, -1.01f);
                pc.Glass_tube = false;
                pc.count++;
            }
            else if (transform.gameObject.name.Equals("Water Basin") && pc.Water_Basin)
            {
                transform.position = new Vector3(0.23f,1.0f, -0.88f);
                pc.Water_Basin = false;
                pc.count++;
            }
            else if (transform.gameObject.name.Equals("Glass funnel") && pc.Glass_funnel)
            {
                transform.position = new Vector3(-0.72f, 1.320955f, -0.84f);
                pc.Glass_funnel = false;
                pc.count++;
            }
            else if (transform.gameObject.name.Equals("Tester") && pc.Tester)
            {
                transform.position = new Vector3(0.19f, 1.26f, -1.08f);
                pc.Tester = false;
                pc.count++;
            }



        }

}
```

Figure 4.2: How to Drag Objects part 2



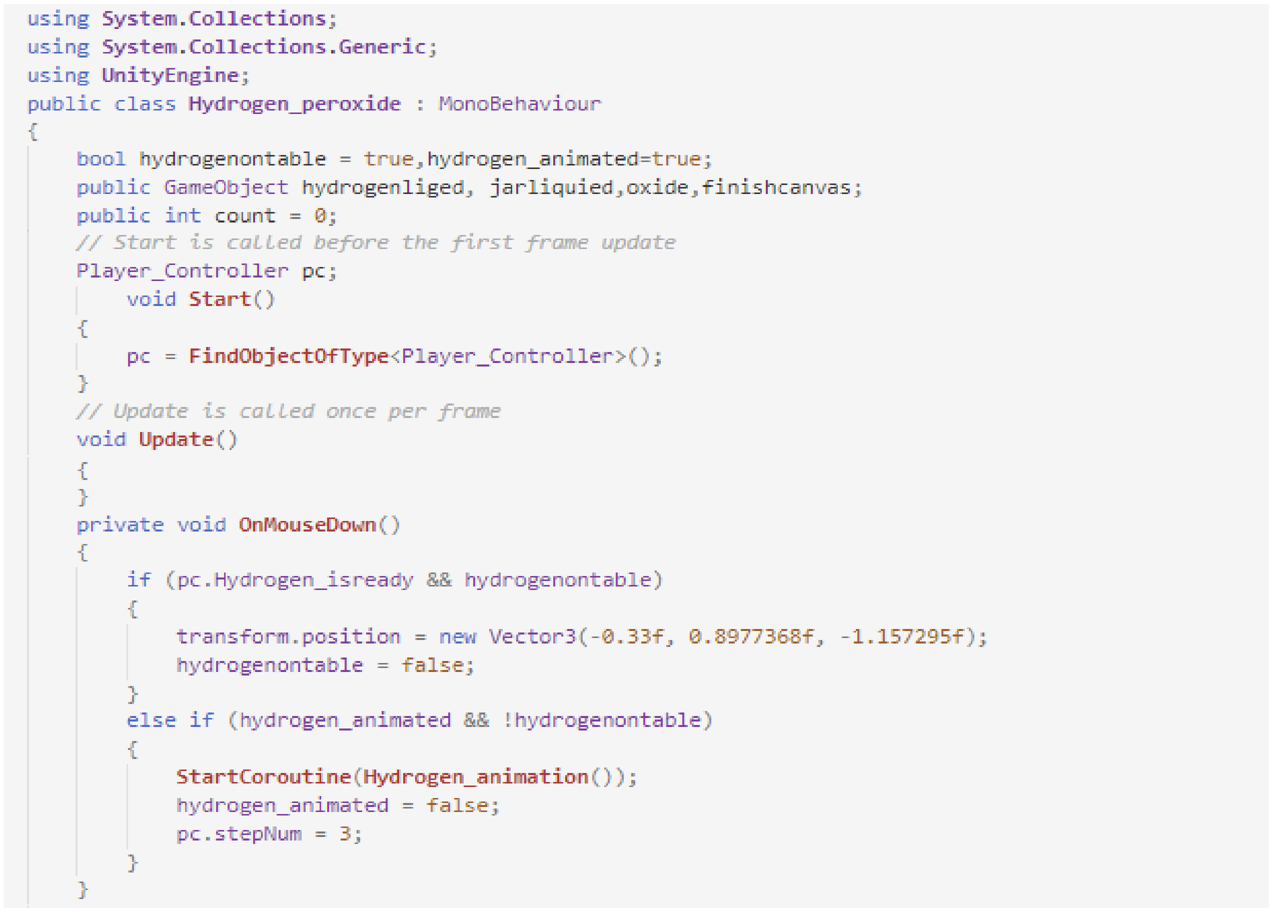
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class EarthClick : MonoBehaviour {
    // Use this for initialization
    void Start () {
    }
    // Update is called once per frame
    void Update () {
    }
    void OnMouseDown() {
        bool zoom = ZoomTarget.zoom;
        if (zoom) {
            ZoomTarget.zoom = false;
            ZoomTarget.target_tag = 0;
            TimeScale.header = "Solar System";
            TimeScale.info = "Mercury\nVenus\nEarth\nMars";
        } else {
            ZoomTarget.zoom = true;
            ZoomTarget.target_tag = 3;
            TimeScale.header = "Earth";
            TimeScale.info = "Mass: 5.97 * 10^24 kg\nDiameter: 12756" +
            "km\nGravity: 9.8 m/s^2\nDistance from Sun: 149.6  * 10^6 km\n\nMoon:\nMass:"+
            " 0.073 * 10^24 kg\nDiameter: 3475 km\nGravity: 1.6 m/s^2\nDistance from Sun: 0.384 * 10^6 km";
        }
    }
```

Figure 4.3: Earth Click



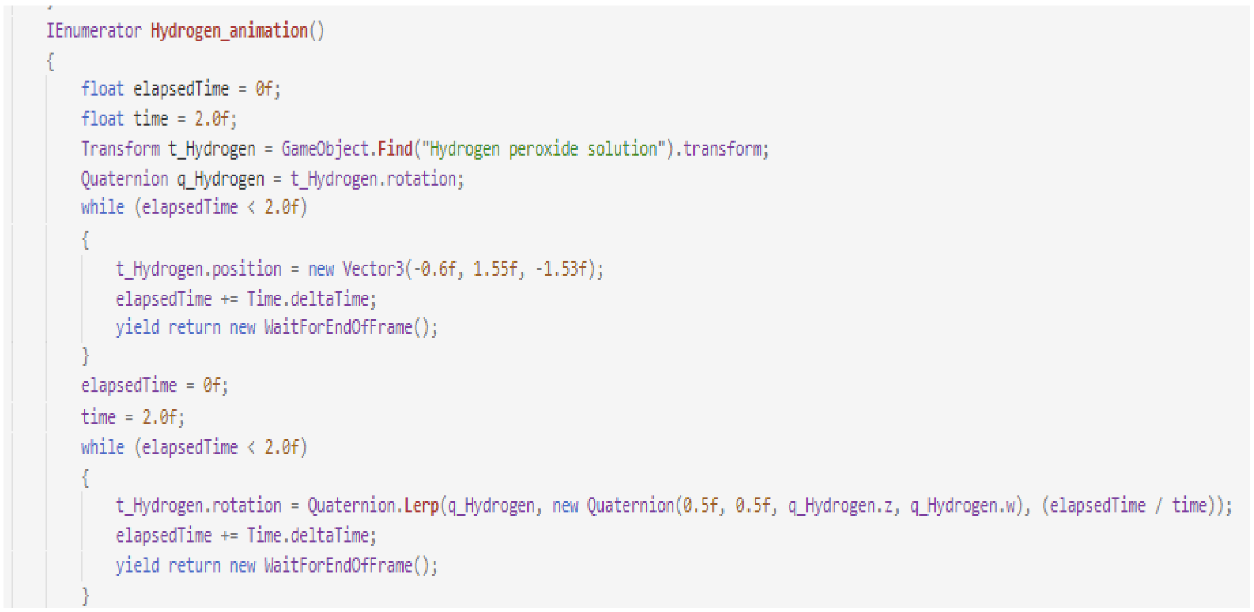
```
using UnityEngine;
using System.Collections;
using UnityEngine.SceneManagement;
public class EarthSystemController : MonoBehaviour {
    public float radius = 34.0f;
    public float speed = 0.5f;
    public float selfSpeed = 12.0f;
    private Transform Earth;
    // Use this for initialization
    void Start () {
        Earth = this.transform.FindChild("Earth");
    }
    // Update is called once per frame
    void Update () {
        Earth.Rotate(Vector3.up, Time.deltaTime * selfSpeed);
       this.transform.localPosition = GetPosition(Time.time * speed);
    }
    private Vector3 GetPosition(float angle)
    {
        return new Vector3(radius * Mathf.Sin(angle), 0, radius * Mathf.Cos(angle));
    }
    void OnMouseDown() {
        bool zoom = ZoomTarget.zoom;
        if (zoom) {
            ZoomTarget.zoom = false;
            ZoomTarget.target_tag = 0;
        } else {
            ZoomTarget.zoom = true;
            ZoomTarget.target_tag = 3;
        }
    }
}
```

Figure 4.4: Earth System Controller



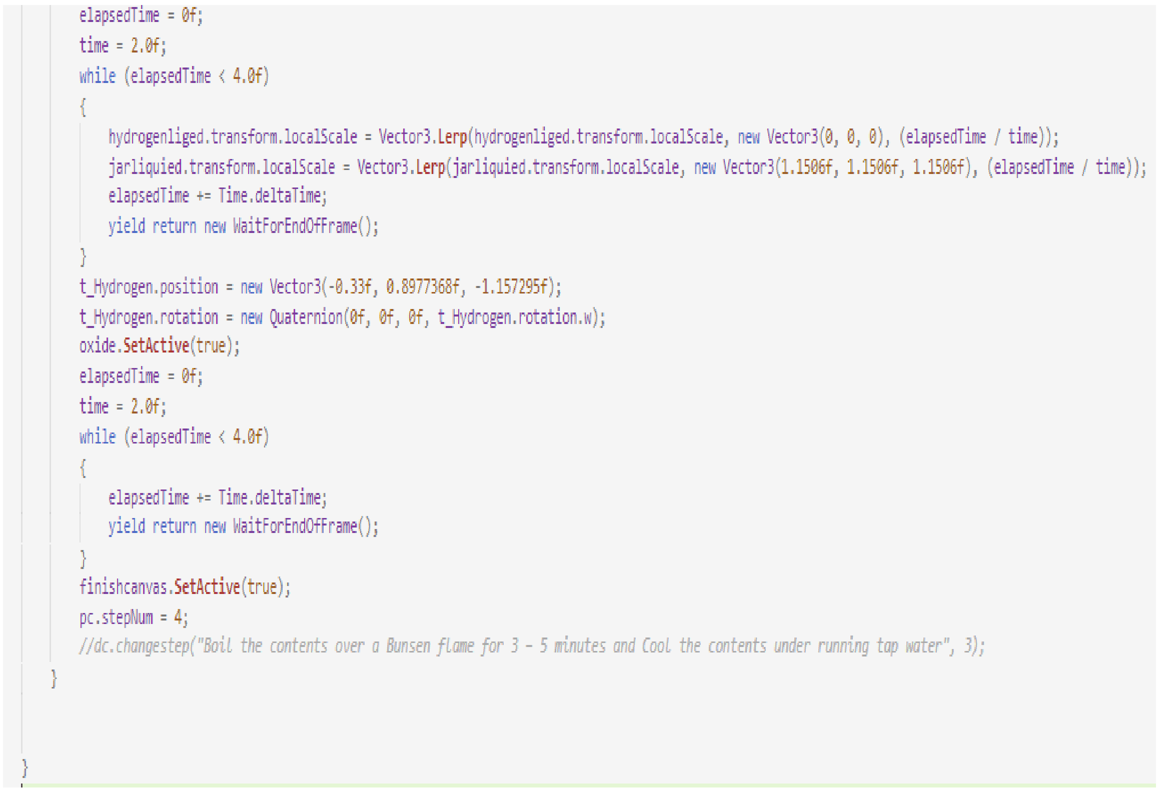
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Hydrogen_peroxide : MonoBehaviour
{
    bool hydrogenontable = true,hydrogen_animated=true;
    public GameObject hydrogenliged, jarliquied,oxide,finishcanvas;
    public int count = 0;
    // Start is called before the first frame update
    Player_Controller pc;
        void Start()
    {
        pc = FindObjectOfType<Player_Controller>();
    }
    // Update is called once per frame
    void Update()
    {
    }
    private void OnMouseDown()
    {
        if (pc.Hydrogen_isready && hydrogenontable)
        {
            transform.position = new Vector3(-0.33f, 0.8977368f, -1.157295f);
            hydrogenontable = false;
        }
        else if (hydrogen_animated && !hydrogenontable)
        {
            StartCoroutine(Hydrogen_animation());
            hydrogen_animated = false;
            pc.stepNum = 3;
        }
    }
```

Figure 4.5: Hydrogen Peroxide part1



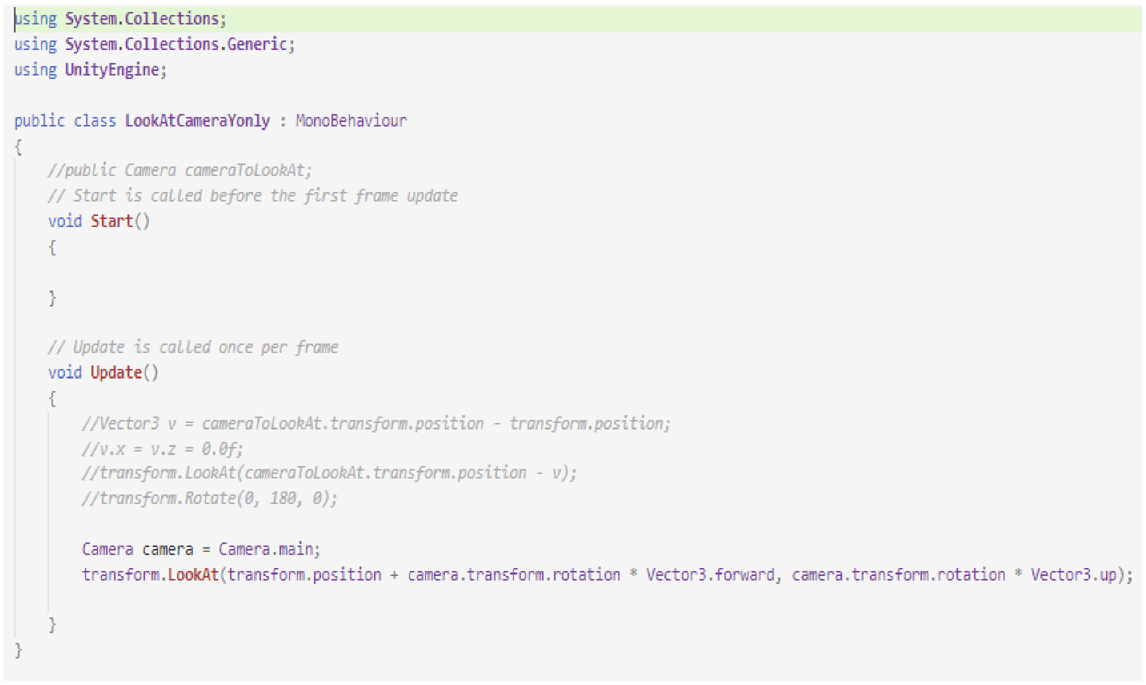
```
    IEnumerator Hydrogen_animation()
    {
        float elapsedTime = 0f;
        float time = 2.0f;
        Transform t_Hydrogen = GameObject.Find("Hydrogen peroxide solution").transform;
        Quaternion q_Hydrogen = t_Hydrogen.rotation;
        while (elapsedTime < 2.0f)
        {
            t_Hydrogen.position = new Vector3(-0.6f, 1.55f, -1.53f);
            elapsedTime += Time.deltaTime;
            yield return new WaitForEndOfFrame();
        }
        elapsedTime = 0f;
        time = 2.0f;
        while (elapsedTime < 2.0f)
        {
            t_Hydrogen.rotation = Quaternion.Lerp(q_Hydrogen, new Quaternion(0.5f, 0.5f, q_Hydrogen.z, q_Hydrogen.w), (elapsedTime / time));
            elapsedTime += Time.deltaTime;
            yield return new WaitForEndOfFrame();
        }
```

Figure 4.6: Hydrogen Peroxide part2



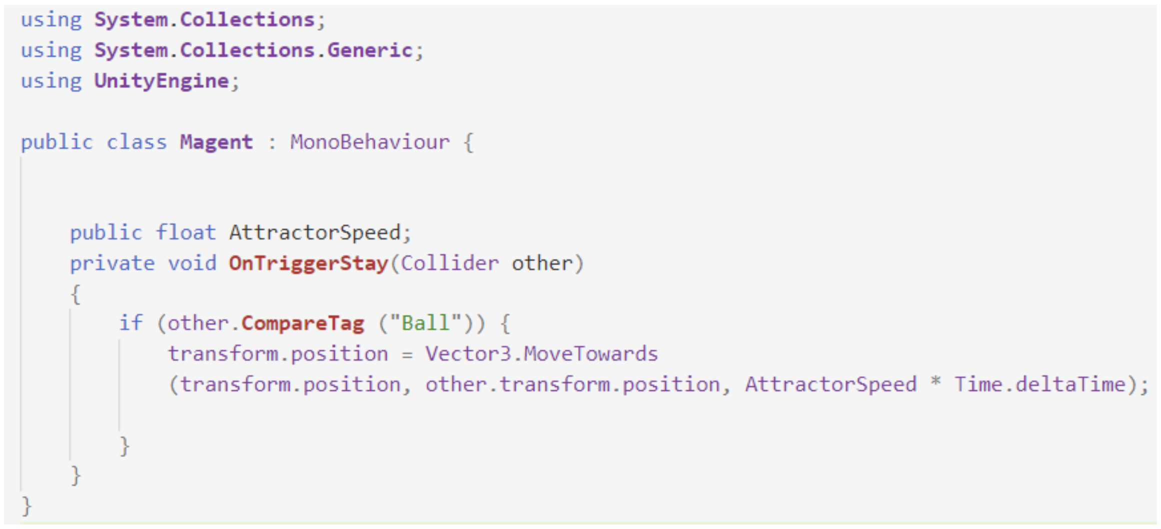
```
        elapsedTime = 0f;
        time = 2.0f;
        while (elapsedTime < 4.0f)
        {
            hydrogenliged.transform.localScale = Vector3.Lerp(hydrogenliged.transform.localScale, new Vector3(0, 0, 0), (elapsedTime / time));
            jarliquied.transform.localScale = Vector3.Lerp(jarliquied.transform.localScale, new Vector3(1.1506f, 1.1506f, 1.1506f), (elapsedTime / time));
            elapsedTime += Time.deltaTime;
            yield return new WaitForEndOfFrame();
        }
        t_Hydrogen.position = new Vector3(-0.33f, 0.8977368f, -1.157295f);
        t_Hydrogen.rotation = new Quaternion(0f, 0f, 0f, t_Hydrogen.rotation.w);
        oxide.SetActive(true);
        elapsedTime = 0f;
        time = 2.0f;
        while (elapsedTime < 4.0f)
        {
            elapsedTime += Time.deltaTime;
            yield return new WaitForEndOfFrame();
        }
        finishcanvas.SetActive(true);
        pc.stepNum = 4;
        //dc.changestep("Boil the contents over a Bunsen flame for 3 - 5 minutes and Cool the contents under running tap water", 3);
    }


}
```

Figure 4.7: Hydrogen Peroxide part3



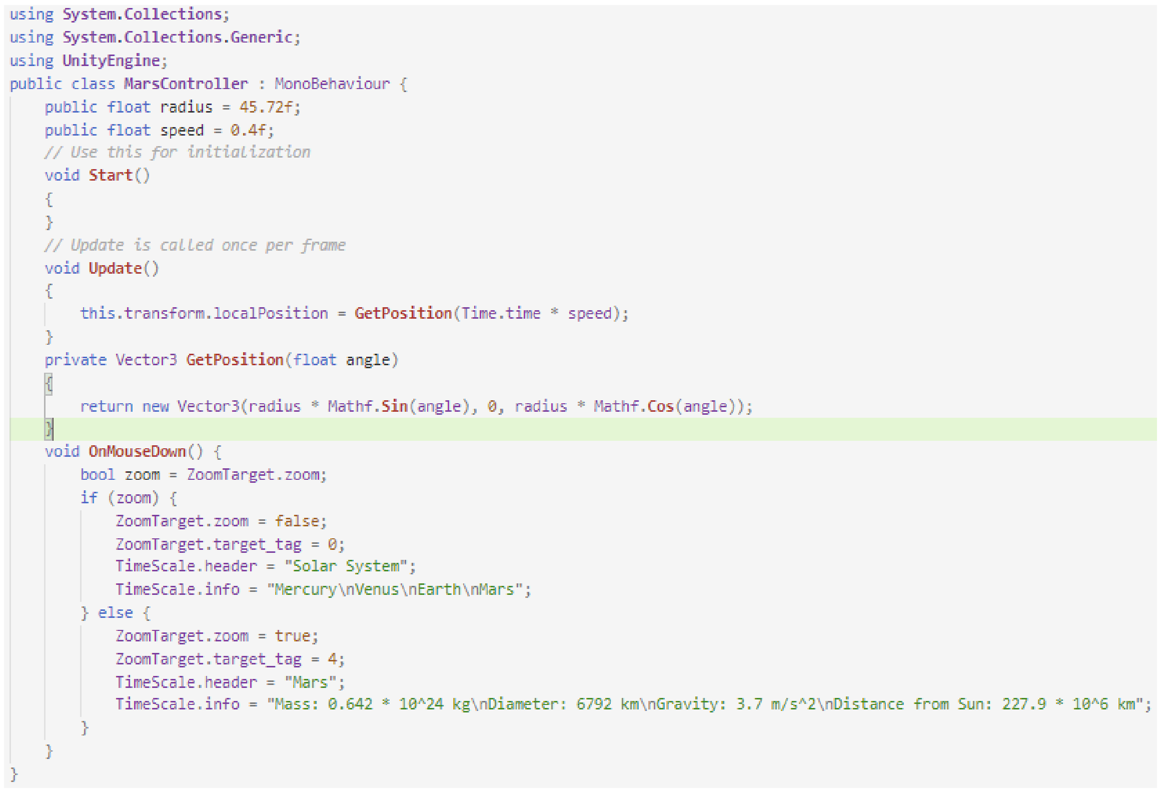
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class LookAtCameraYonly : MonoBehaviour
{
    //public Camera cameraToLookAt;
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        //Vector3 v = cameraToLookAt.transform.position - transform.position;
        //v.x = v.z = 0.0f;
        //transform.LookAt(cameraToLookAt.transform.position - v);
        //transform.Rotate(0, 180, 0);

        Camera camera = Camera.main;
        transform.LookAt(transform.position + camera.transform.rotation * Vector3.forward, camera.transform.rotation * Vector3.up);

    }
}
```

Figure 4.8: Look at Camera y only



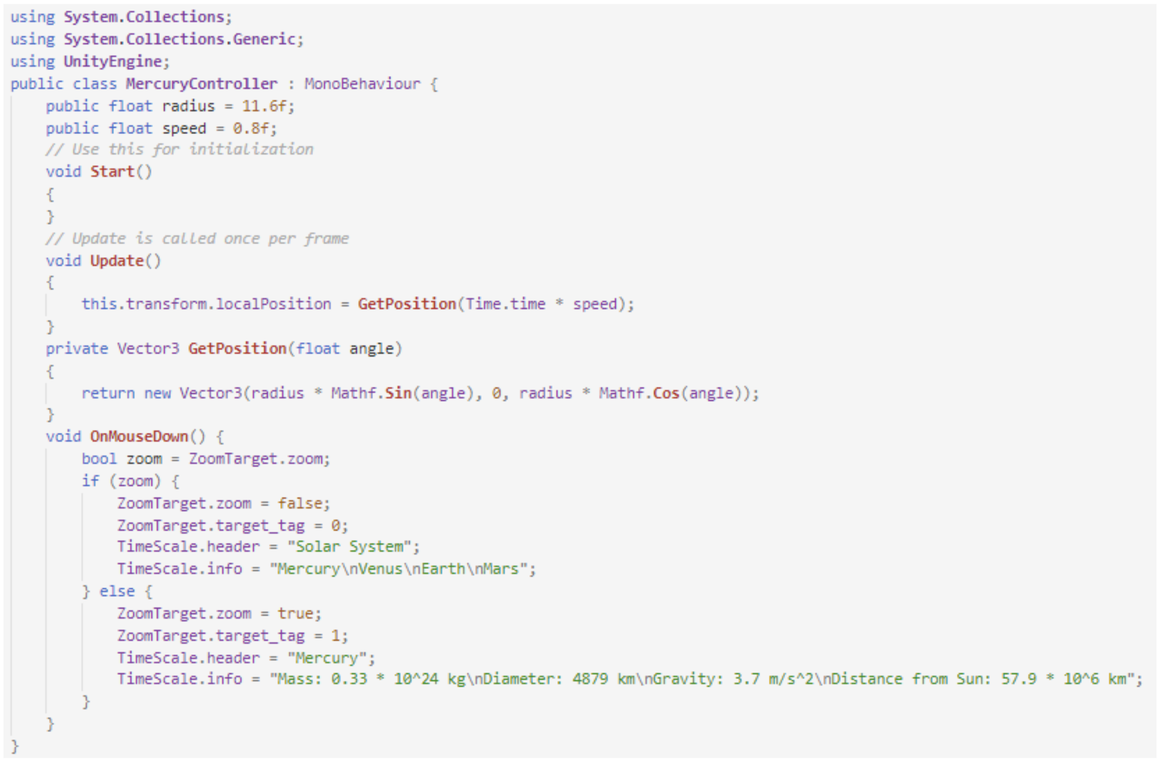
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Magent : MonoBehaviour {


    public float AttractorSpeed;
    private void OnTriggerStay(Collider other)
    {
        if (other.CompareTag ("Ball")) {
            transform.position = Vector3.MoveTowards
            (transform.position, other.transform.position, AttractorSpeed * Time.deltaTime);

        }
    }
}
```

Figure 4.9: Magnet



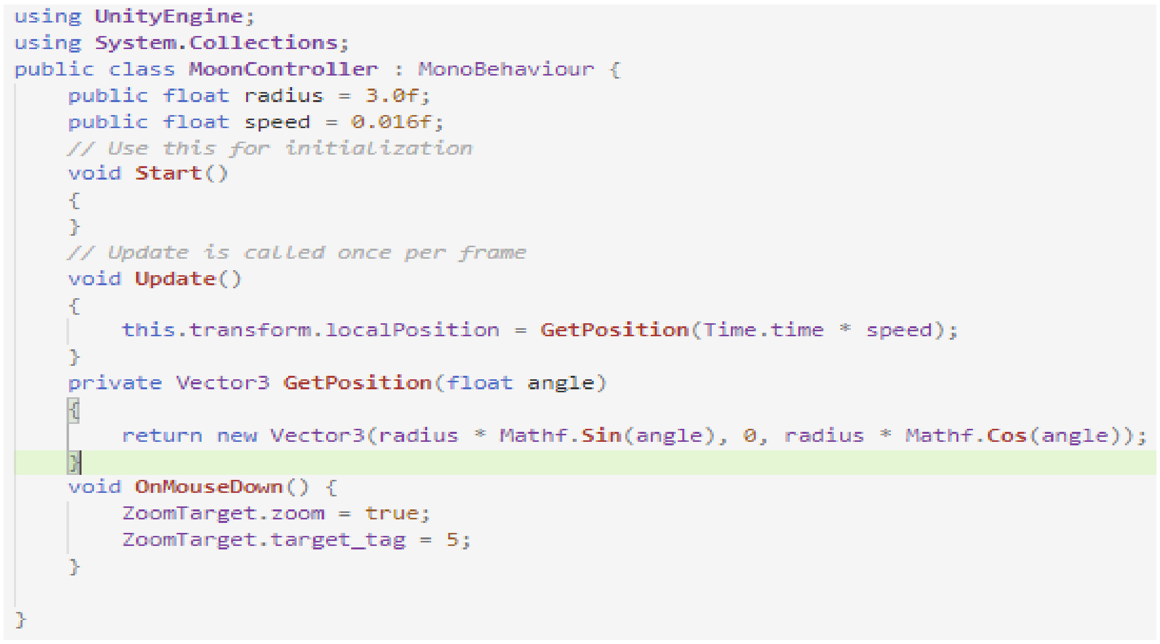
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class MarsController : MonoBehaviour {
    public float radius = 45.72f;
    public float speed = 0.4f;
    // Use this for initialization
    void Start()
    {
    }
    // Update is called once per frame
    void Update()
    {
        this.transform.localPosition = GetPosition(Time.time * speed);
    }
    private Vector3 GetPosition(float angle)
    {
        return new Vector3(radius * Mathf.Sin(angle), 0, radius * Mathf.Cos(angle));
    }
    void OnMouseDown() {
        bool zoom = ZoomTarget.zoom;
        if (zoom) {
            ZoomTarget.zoom = false;
            ZoomTarget.target_tag = 0;
            TimeScale.header = "Solar System";
            TimeScale.info = "Mercury\nVenus\nEarth\nMars";
        } else {
            ZoomTarget.zoom = true;
            ZoomTarget.target_tag = 4;
            TimeScale.header = "Mars";
            TimeScale.info = "Mass: 0.642 * 10^24 kg\nDiameter: 6792 km\nGravity: 3.7 m/s^2\nDistance from Sun: 227.9 * 10^6 km";
        }
    }
}
```

Figure 4.10: Mars Controller



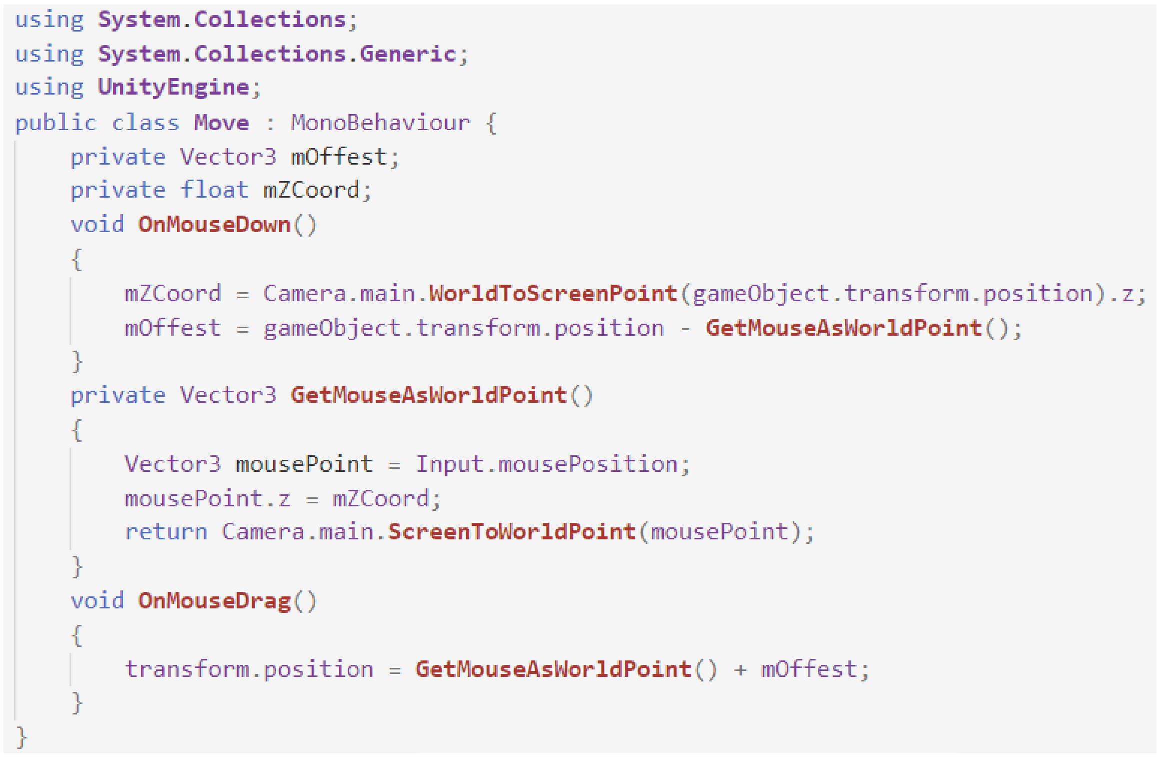
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class MercuryController : MonoBehaviour {
    public float radius = 11.6f;
    public float speed = 0.8f;
    // Use this for initialization
    void Start()
    {
    }
    // Update is called once per frame
    void Update()
    {
        this.transform.localPosition = GetPosition(Time.time * speed);
    }
    private Vector3 GetPosition(float angle)
    {
        return new Vector3(radius * Mathf.Sin(angle), 0, radius * Mathf.Cos(angle));
    }
    void OnMouseDown() {
        bool zoom = ZoomTarget.zoom;
        if (zoom) {
            ZoomTarget.zoom = false;
            ZoomTarget.target_tag = 0;
            TimeScale.header = "Solar System";
            TimeScale.info = "Mercury\nVenus\nEarth\nMars";
        } else {
            ZoomTarget.zoom = true;
            ZoomTarget.target_tag = 1;
            TimeScale.header = "Mercury";
            TimeScale.info = "Mass: 0.33 * 10^24 kg\nDiameter: 4879 km\nGravity: 3.7 m/s^2\nDistance from Sun: 57.9 * 10^6 km";
        }
    }
}
```

Figure 4.11: Mercury Controller



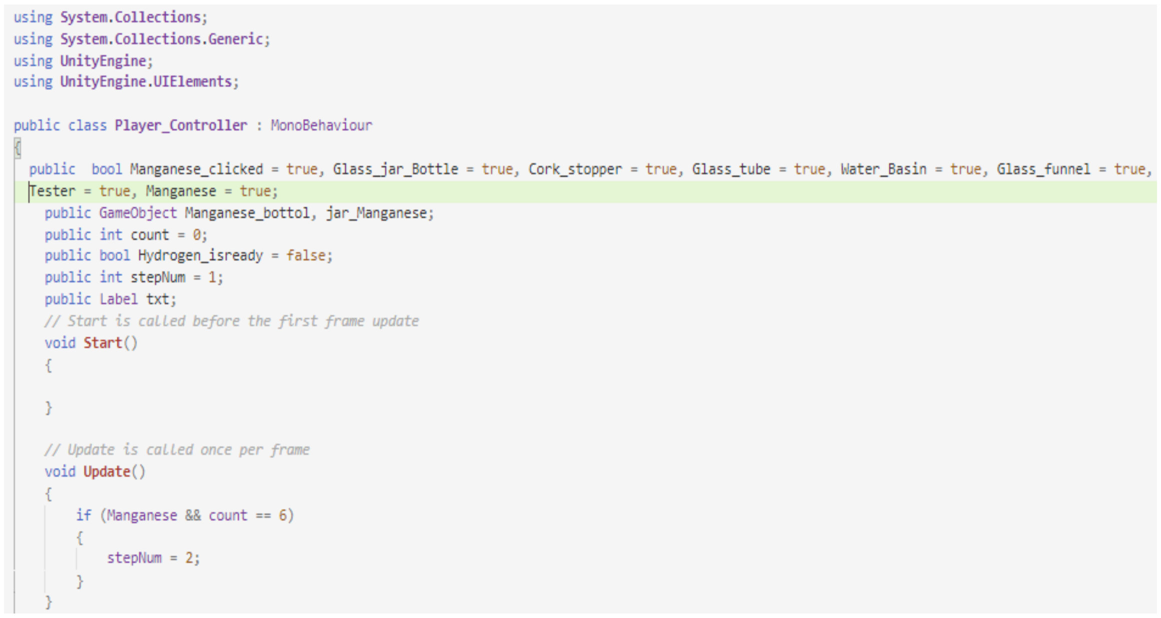
```
using UnityEngine;
using System.Collections;
public class MoonController : MonoBehaviour {
    public float radius = 3.0f;
    public float speed = 0.016f;
    // Use this for initialization
    void Start()
    {
    }
    // Update is called once per frame
    void Update()
    {
        this.transform.localPosition = GetPosition(Time.time * speed);
    }
    private Vector3 GetPosition(float angle)
    {
        return new Vector3(radius * Mathf.Sin(angle), 0, radius * Mathf.Cos(angle));
    }
    void OnMouseDown() {
        ZoomTarget.zoom = true;
        ZoomTarget.target_tag = 5;
    }

}
```

Figure 4.12: Moon Controller



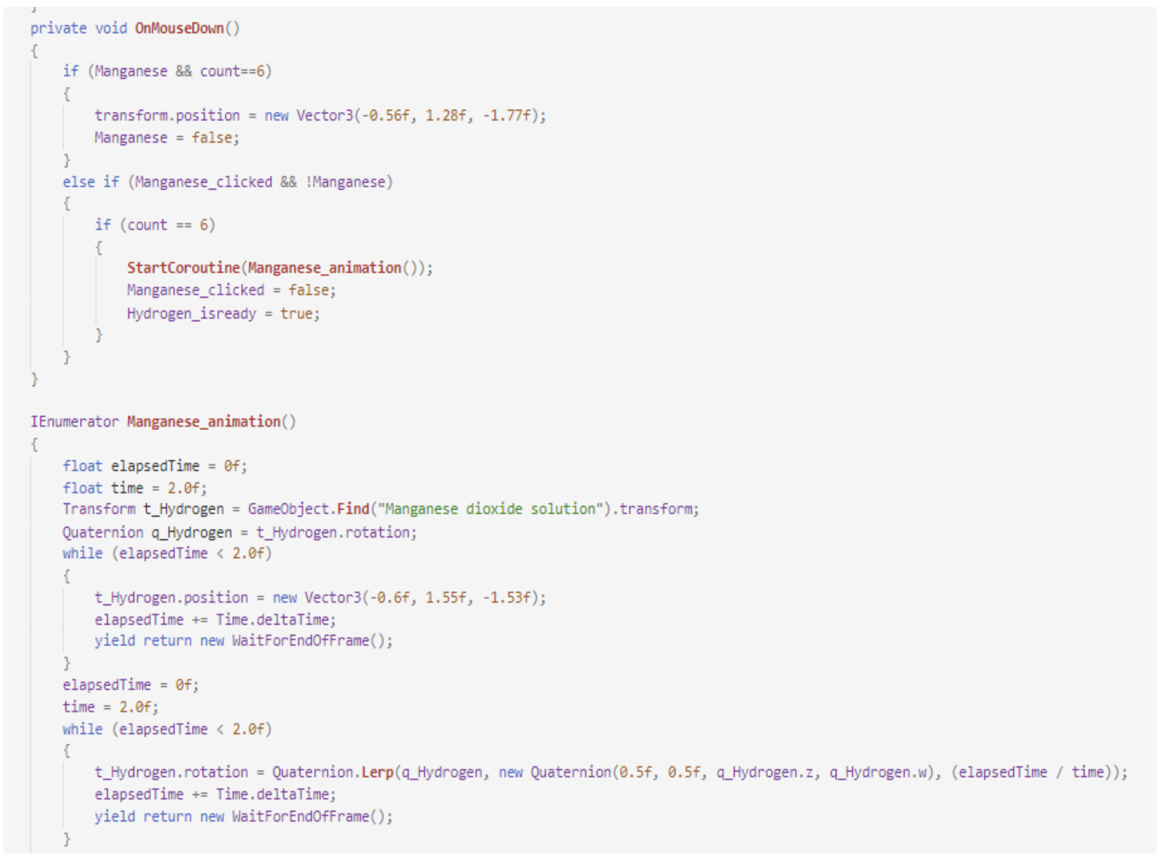
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Move : MonoBehaviour {
    private Vector3 mOffest;
    private float mZCoord;
    void OnMouseDown()
    {
        mZCoord = Camera.main.WorldToScreenPoint(gameObject.transform.position).z;
        mOffest = gameObject.transform.position - GetMouseAsWorldPoint();
    }
    private Vector3 GetMouseAsWorldPoint()
    {
        Vector3 mousePoint = Input.mousePosition;
        mousePoint.z = mZCoord;
        return Camera.main.ScreenToWorldPoint(mousePoint);
    }
    void OnMouseDrag()
    {
        transform.position = GetMouseAsWorldPoint() + mOffest;
    }
}
```

Figure 4.13: Move



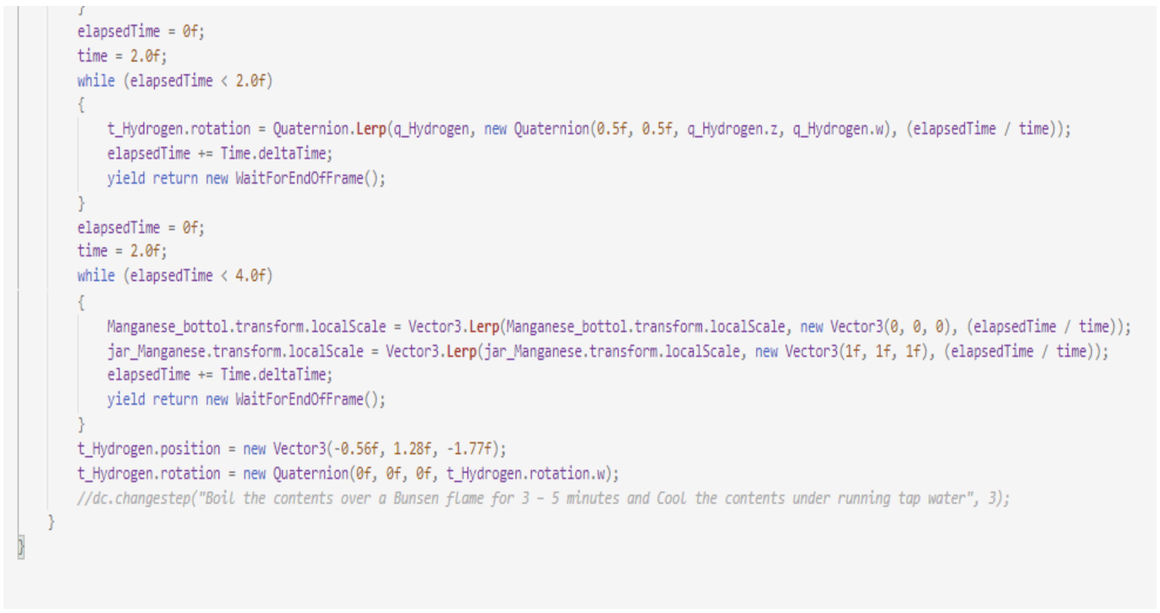
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UIElements;

public class Player_Controller : MonoBehaviour
{
  public  bool Manganese_clicked = true, Glass_jar_Bottle = true, Cork_stopper = true, Glass_tube = true, Water_Basin = true, Glass_funnel = true,
 Tester = true, Manganese = true;
    public GameObject Manganese_bottol, jar_Manganese;
    public int count = 0;
    public bool Hydrogen_isready = false;
    public int stepNum = 1;
    public Label txt;
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        if (Manganese && count == 6)
        {
            stepNum = 2;
        }
    }
```

Figure 4.14: Player Controller part 1



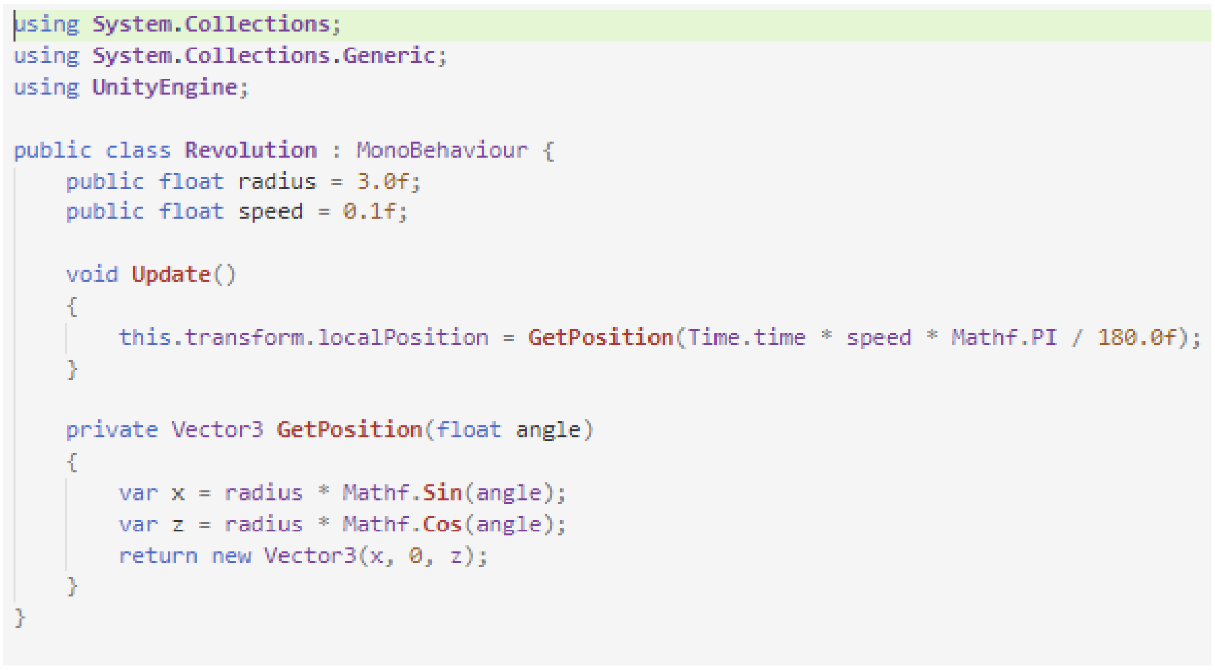
```
private void OnMouseDown()
{
    if (Manganese && count==6)
    {
        transform.position = new Vector3(-0.56f, 1.28f, -1.77f);
        Manganese = false;
    }
    else if (Manganese_clicked && !Manganese)
    {
        if (count == 6)
        {
            StartCoroutine(Manganese_animation());
            Manganese_clicked = false;
            Hydrogen_isready = true;
        }
    }
}

IEnumerator Manganese_animation()
{
    float elapsedTime = 0f;
    float time = 2.0f;
    Transform t_Hydrogen = GameObject.Find("Manganese dioxide solution").transform;
    Quaternion q_Hydrogen = t_Hydrogen.rotation;
    while (elapsedTime < 2.0f)
    {
        t_Hydrogen.position = new Vector3(-0.6f, 1.55f, -1.53f);
        elapsedTime += Time.deltaTime;
        yield return new WaitForEndOfFrame();
    }
    elapsedTime = 0f;
    time = 2.0f;
    while (elapsedTime < 2.0f)
    {
        t_Hydrogen.rotation = Quaternion.Lerp(q_Hydrogen, new Quaternion(0.5f, 0.5f, q_Hydrogen.z, q_Hydrogen.w), (elapsedTime / time));
        elapsedTime += Time.deltaTime;
        yield return new WaitForEndOfFrame();
    }
```

Figure 4.15: Player Controller part 2



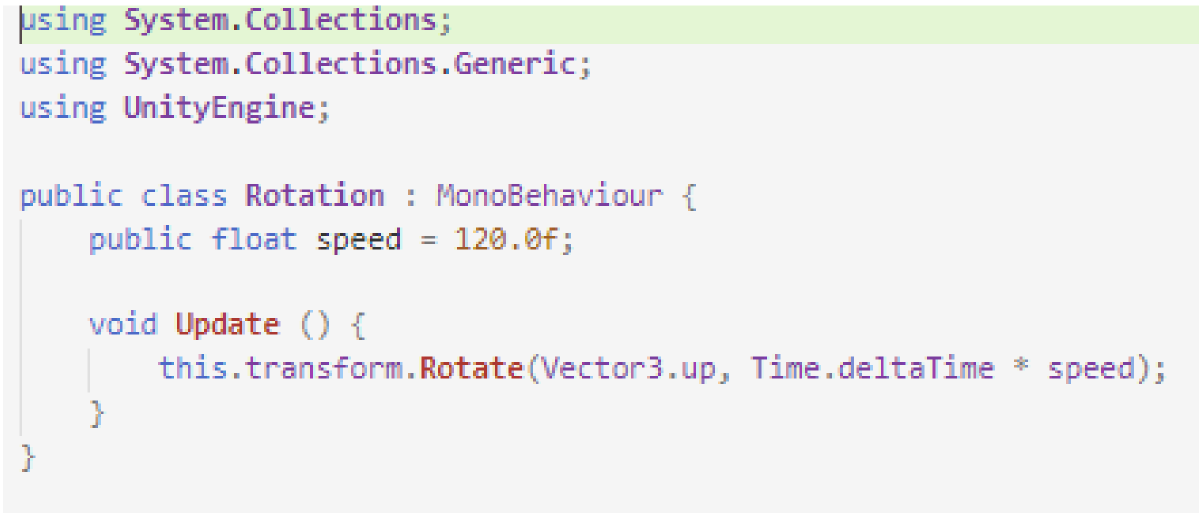
```
        elapsedTime = 0f;
        time = 2.0f;
        while (elapsedTime < 2.0f)
        {
            t_Hydrogen.rotation = Quaternion.Lerp(q_Hydrogen, new Quaternion(0.5f, 0.5f, q_Hydrogen.z, q_Hydrogen.w), (elapsedTime / time));
            elapsedTime += Time.deltaTime;
            yield return new WaitForEndOfFrame();
        }
        elapsedTime = 0f;
        time = 2.0f;
        while (elapsedTime < 4.0f)
        {
            Manganese_bottol.transform.localScale = Vector3.Lerp(Manganese_bottol.transform.localScale, new Vector3(0, 0, 0), (elapsedTime / time));
            jar_Manganese.transform.localScale = Vector3.Lerp(jar_Manganese.transform.localScale, new Vector3(1f, 1f, 1f), (elapsedTime / time));
            elapsedTime += Time.deltaTime;
            yield return new WaitForEndOfFrame();
        }
        t_Hydrogen.position = new Vector3(-0.56f, 1.28f, -1.77f);
        t_Hydrogen.rotation = new Quaternion(0f, 0f, 0f, t_Hydrogen.rotation.w);
        //dc.changestep("Boil the contents over a Bunsen flame for 3 - 5 minutes and Cool the contents under running tap water", 3);
    }
}
```

Figure 4.16: Player Controller part 3



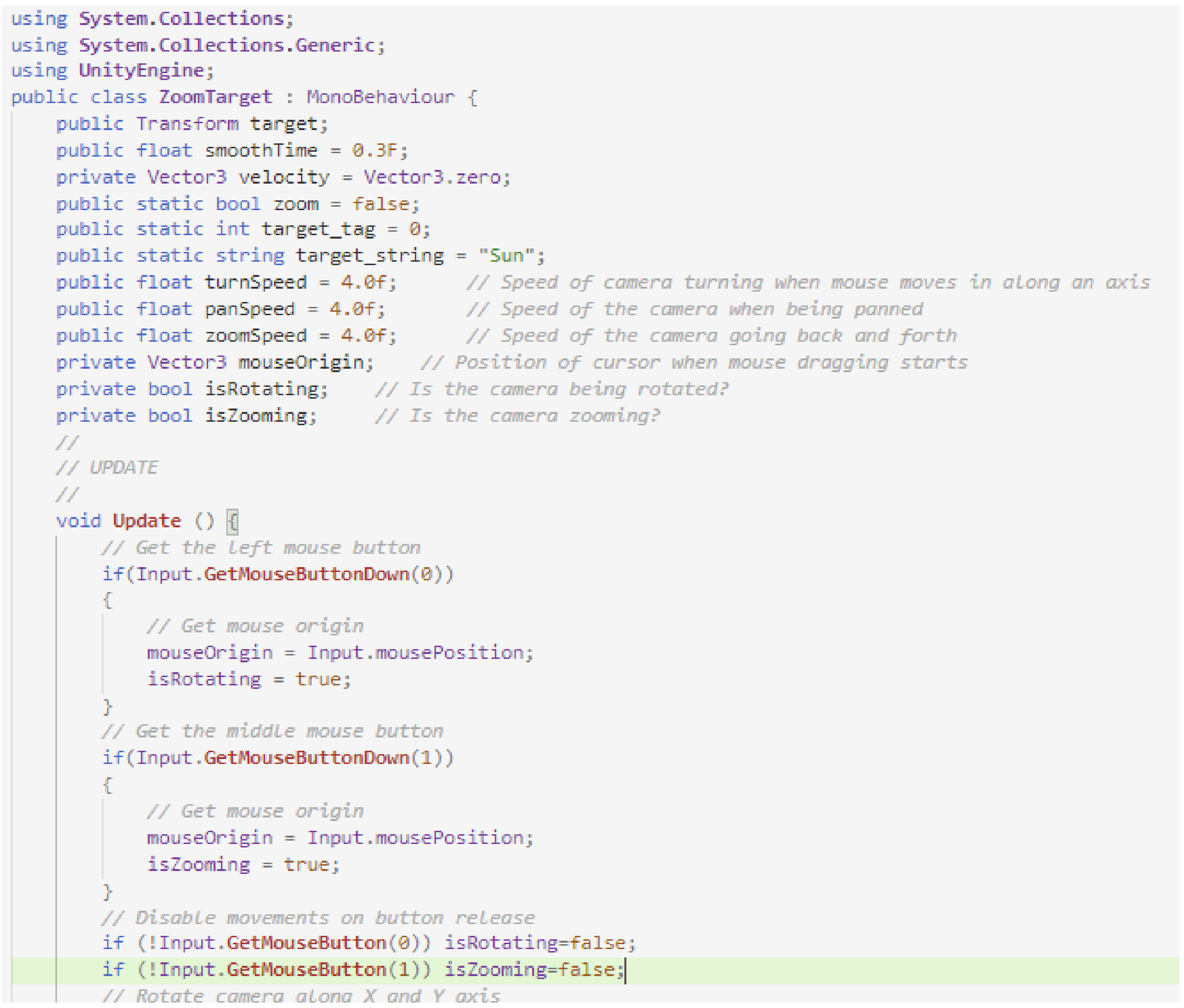
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Revolution : MonoBehaviour {
    public float radius = 3.0f;
    public float speed = 0.1f;

    void Update()
    {
        this.transform.localPosition = GetPosition(Time.time * speed * Mathf.PI / 180.0f);
    }

    private Vector3 GetPosition(float angle)
    {
        var x = radius * Mathf.Sin(angle);
        var z = radius * Mathf.Cos(angle);
        return new Vector3(x, 0, z);
    }
}
```

Figure 4.17: Revolution



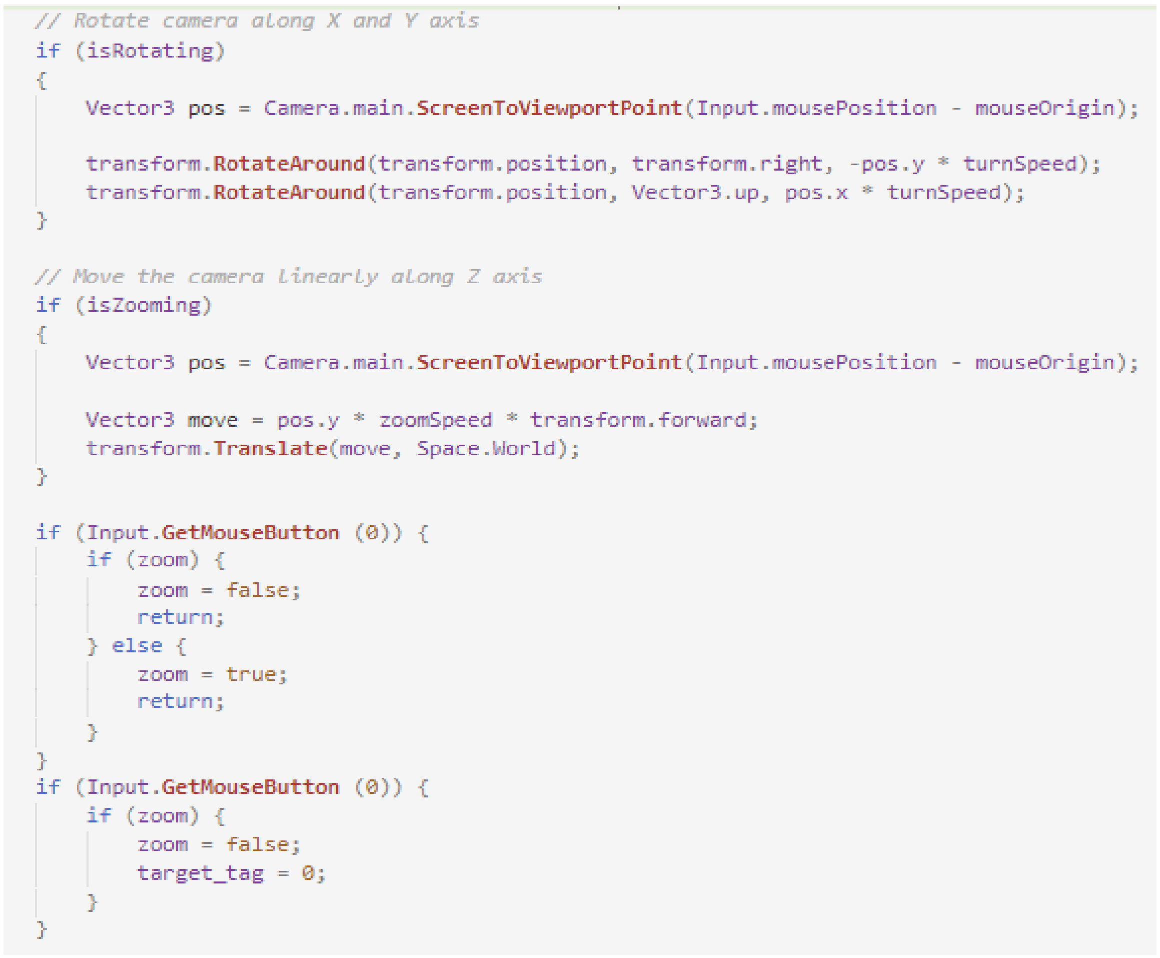
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Rotation : MonoBehaviour {
    public float speed = 120.0f;

    void Update () {
        this.transform.Rotate(Vector3.up, Time.deltaTime * speed);
    }
}
```

Figure 4.18: Rotation



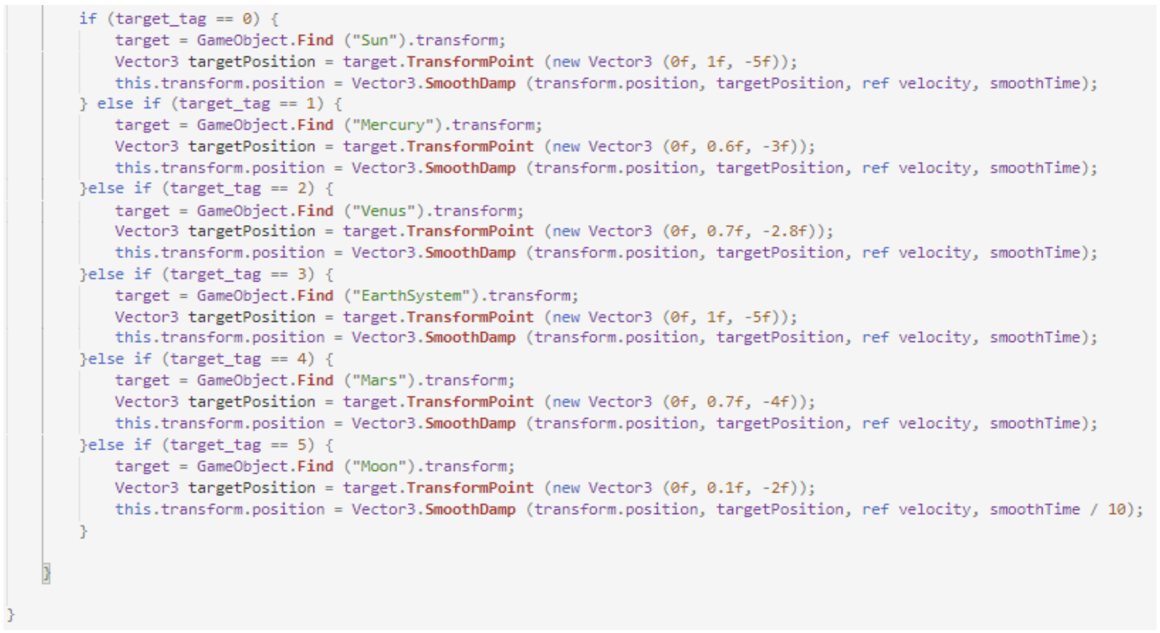
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class ZoomTarget : MonoBehaviour {
    public Transform target;
    public float smoothTime = 0.3F;
    private Vector3 velocity = Vector3.zero;
    public static bool zoom = false;
    public static int target_tag = 0;
    public static string target_string = "Sun";
    public float turnSpeed = 4.0f;      // Speed of camera turning when mouse moves in along an axis
    public float panSpeed = 4.0f;       // Speed of the camera when being panned
    public float zoomSpeed = 4.0f;      // Speed of the camera going back and forth
    private Vector3 mouseOrigin;    // Position of cursor when mouse dragging starts
    private bool isRotating;    // Is the camera being rotated?
    private bool isZooming;     // Is the camera zooming?
    //
    // UPDATE
    //
    void Update () {
        // Get the left mouse button
        if(Input.GetMouseButtonDown(0))
        {
            // Get mouse origin
            mouseOrigin = Input.mousePosition;
            isRotating = true;
        }
        // Get the middle mouse button
        if(Input.GetMouseButtonDown(1))
        {
            // Get mouse origin
            mouseOrigin = Input.mousePosition;
            isZooming = true;
        }
        // Disable movements on button release
        if (!Input.GetMouseButton(0)) isRotating=false;
        if (!Input.GetMouseButton(1)) isZooming=false;
        // Rotate camera along X and Y axis
```

Figure 4.19: Zoom Target part 1



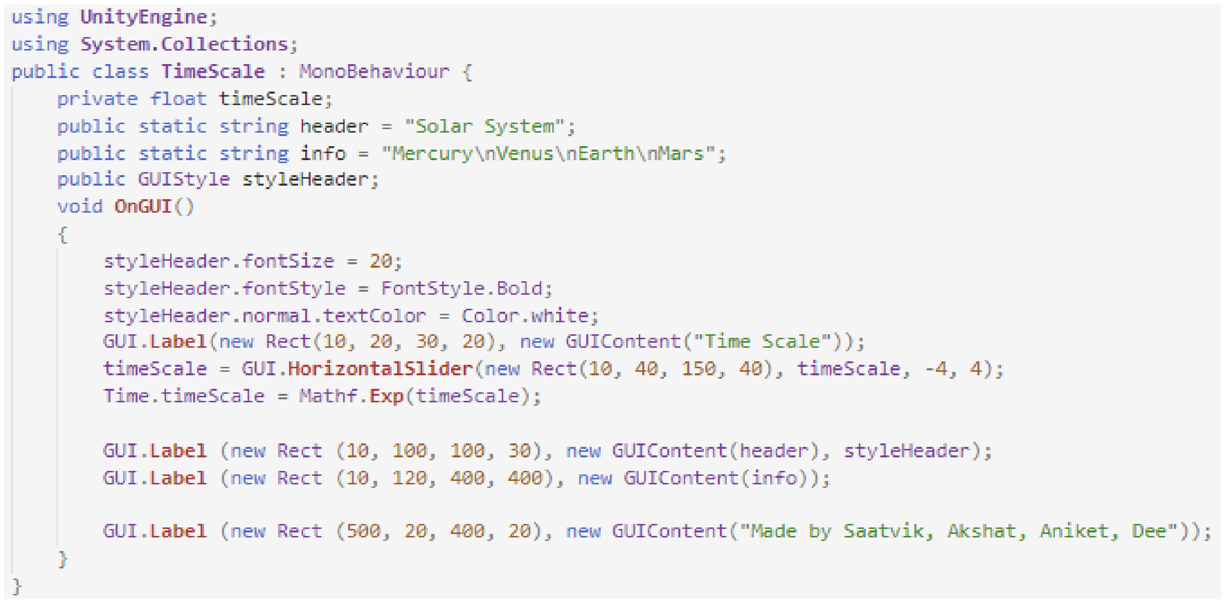
```
// Rotate camera along X and Y axis
if (isRotating)
{
    Vector3 pos = Camera.main.ScreenToViewportPoint(Input.mousePosition - mouseOrigin);

    transform.RotateAround(transform.position, transform.right, -pos.y * turnSpeed);
    transform.RotateAround(transform.position, Vector3.up, pos.x * turnSpeed);
}

// Move the camera linearly along Z axis
if (isZooming)
{
    Vector3 pos = Camera.main.ScreenToViewportPoint(Input.mousePosition - mouseOrigin);

    Vector3 move = pos.y * zoomSpeed * transform.forward;
    transform.Translate(move, Space.World);
}

if (Input.GetMouseButton (0)) {
    if (zoom) {
        zoom = false;
        return;
    } else {
        zoom = true;
        return;
    }
}
if (Input.GetMouseButton (0)) {
    if (zoom) {
        zoom = false;
        target_tag = 0;
    }
}
```

Figure 4.20: Zoom Target part 2



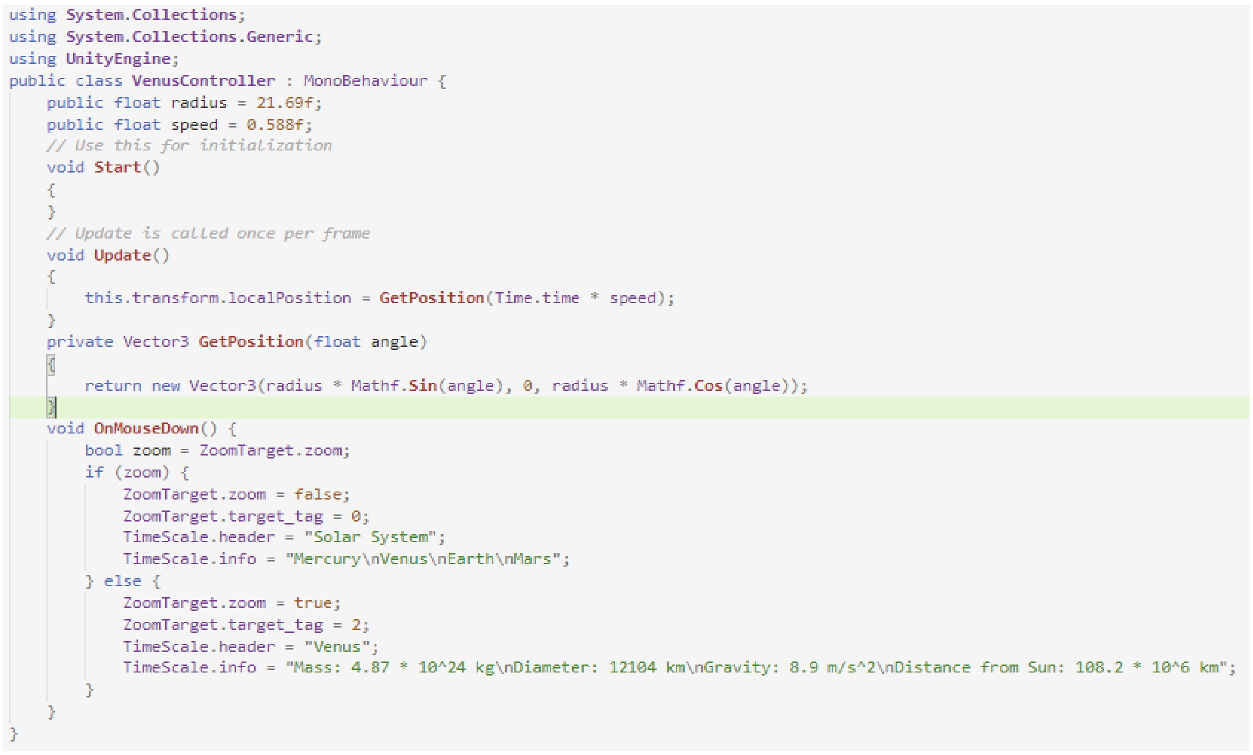
```
        if (target_tag == 0) {
            target = GameObject.Find ("Sun").transform;
            Vector3 targetPosition = target.TransformPoint (new Vector3 (0f, 1f, -5f));
            this.transform.position = Vector3.SmoothDamp (transform.position, targetPosition, ref velocity, smoothTime);
        } else if (target_tag == 1) {
            target = GameObject.Find ("Mercury").transform;
            Vector3 targetPosition = target.TransformPoint (new Vector3 (0f, 0.6f, -3f));
            this.transform.position = Vector3.SmoothDamp (transform.position, targetPosition, ref velocity, smoothTime);
        }else if (target_tag == 2) {
            target = GameObject.Find ("Venus").transform;
            Vector3 targetPosition = target.TransformPoint (new Vector3 (0f, 0.7f, -2.8f));
            this.transform.position = Vector3.SmoothDamp (transform.position, targetPosition, ref velocity, smoothTime);
        }else if (target_tag == 3) {
            target = GameObject.Find ("EarthSystem").transform;
            Vector3 targetPosition = target.TransformPoint (new Vector3 (0f, 1f, -5f));
            this.transform.position = Vector3.SmoothDamp (transform.position, targetPosition, ref velocity, smoothTime);
        }else if (target_tag == 4) {
            target = GameObject.Find ("Mars").transform;
            Vector3 targetPosition = target.TransformPoint (new Vector3 (0f, 0.7f, -4f));
            this.transform.position = Vector3.SmoothDamp (transform.position, targetPosition, ref velocity, smoothTime);
        }else if (target_tag == 5) {
            target = GameObject.Find ("Moon").transform;
            Vector3 targetPosition = target.TransformPoint (new Vector3 (0f, 0.1f, -2f));
            this.transform.position = Vector3.SmoothDamp (transform.position, targetPosition, ref velocity, smoothTime / 10);
        }

    }

}
```

Figure 4.21: Zoom Target part 3



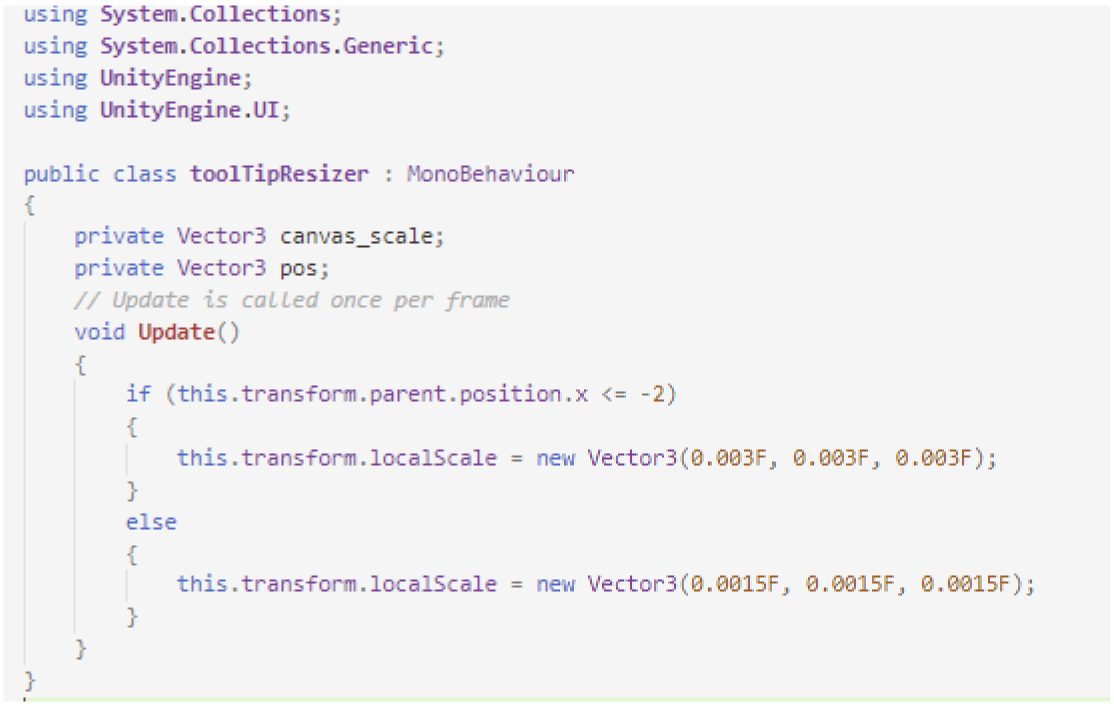
```
using UnityEngine;
using System.Collections;
public class TimeScale : MonoBehaviour {
    private float timeScale;
    public static string header = "Solar System";
    public static string info = "Mercury\nVenus\nEarth\nMars";
    public GUIStyle styleHeader;
    void OnGUI()
    {
        styleHeader.fontSize = 20;
        styleHeader.fontStyle = FontStyle.Bold;
        styleHeader.normal.textColor = Color.white;
        GUI.Label(new Rect(10, 20, 30, 20), new GUIContent("Time Scale"));
        timeScale = GUI.HorizontalSlider(new Rect(10, 40, 150, 40), timeScale, -4, 4);
        Time.timeScale = Mathf.Exp(timeScale);

        GUI.Label (new Rect (10, 100, 100, 30), new GUIContent(header), styleHeader);
        GUI.Label (new Rect (10, 120, 400, 400), new GUIContent(info));

        GUI.Label (new Rect (500, 20, 400, 20), new GUIContent("Made by Saatvik, Akshat, Aniket, Dee"));
    }
}
```

Figure 4.22: Time Scale



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class VenusController : MonoBehaviour {
    public float radius = 21.69f;
    public float speed = 0.588f;
    // Use this for initialization
    void Start()
    {
    }
    // Update is called once per frame
    void Update()
    {
        this.transform.localPosition = GetPosition(Time.time * speed);
    }
    private Vector3 GetPosition(float angle)
    {
        return new Vector3(radius * Mathf.Sin(angle), 0, radius * Mathf.Cos(angle));
    }
    void OnMouseDown() {
        bool zoom = ZoomTarget.zoom;
        if (zoom) {
            ZoomTarget.zoom = false;
            ZoomTarget.target_tag = 0;
            TimeScale.header = "Solar System";
            TimeScale.info = "Mercury\nVenus\nEarth\nMars";
        } else {
            ZoomTarget.zoom = true;
            ZoomTarget.target_tag = 2;
            TimeScale.header = "Venus";
            TimeScale.info = "Mass: 4.87 * 10^24 kg\nDiameter: 12104 km\nGravity: 8.9 m/s^2\nDistance from Sun: 108.2 * 10^6 km";
        }
    }
}
```

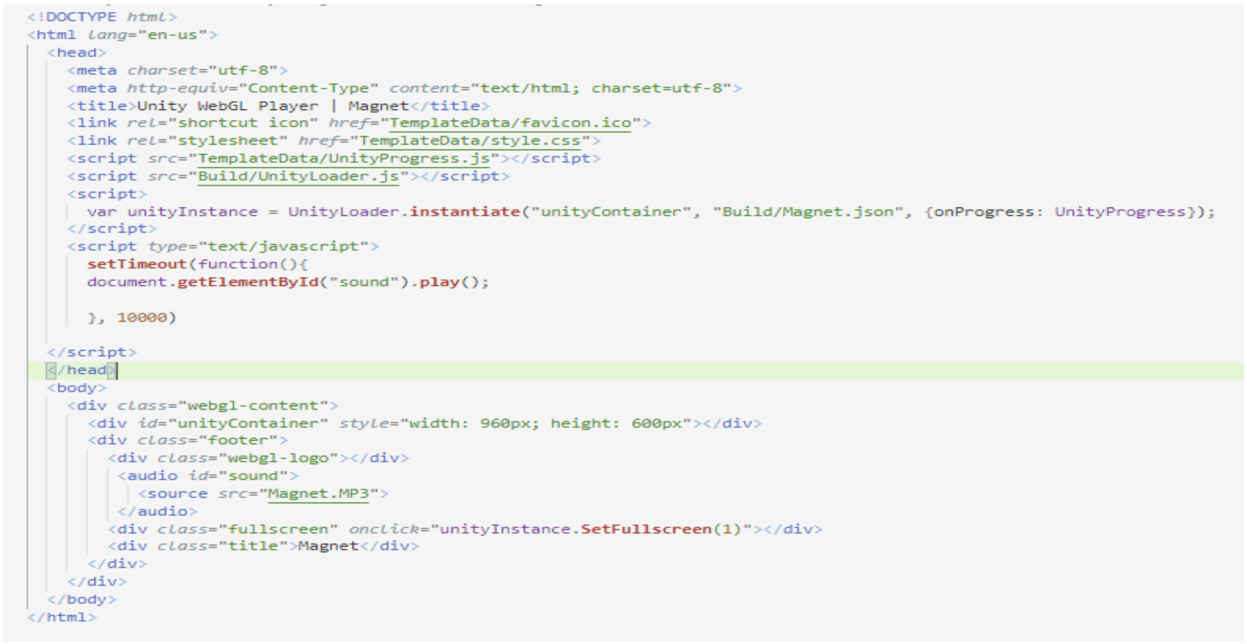
Figure 4.23: Venus Controller



```csharp
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class toolTipResizer : MonoBehaviour
{
    private Vector3 canvas_scale;
    private Vector3 pos;
    // Update is called once per frame
    void Update()
    {
        if (this.transform.parent.position.x <= -2)
        {
            this.transform.localScale = new Vector3(0.003F, 0.003F, 0.003F);
        }
        else
        {
            this.transform.localScale = new Vector3(0.0015F, 0.0015F, 0.0015F);
        }
    }
}
```

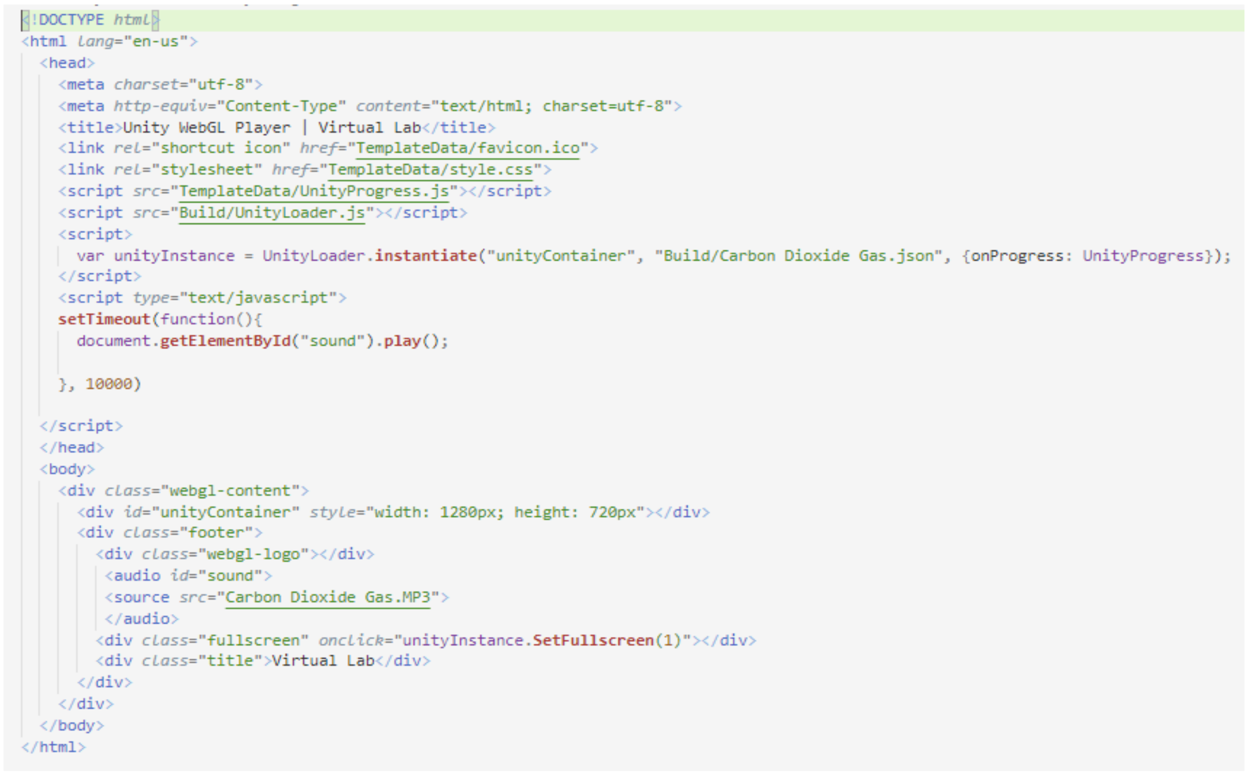
Figure 4.24: Tool Tip Resizer

Frontend



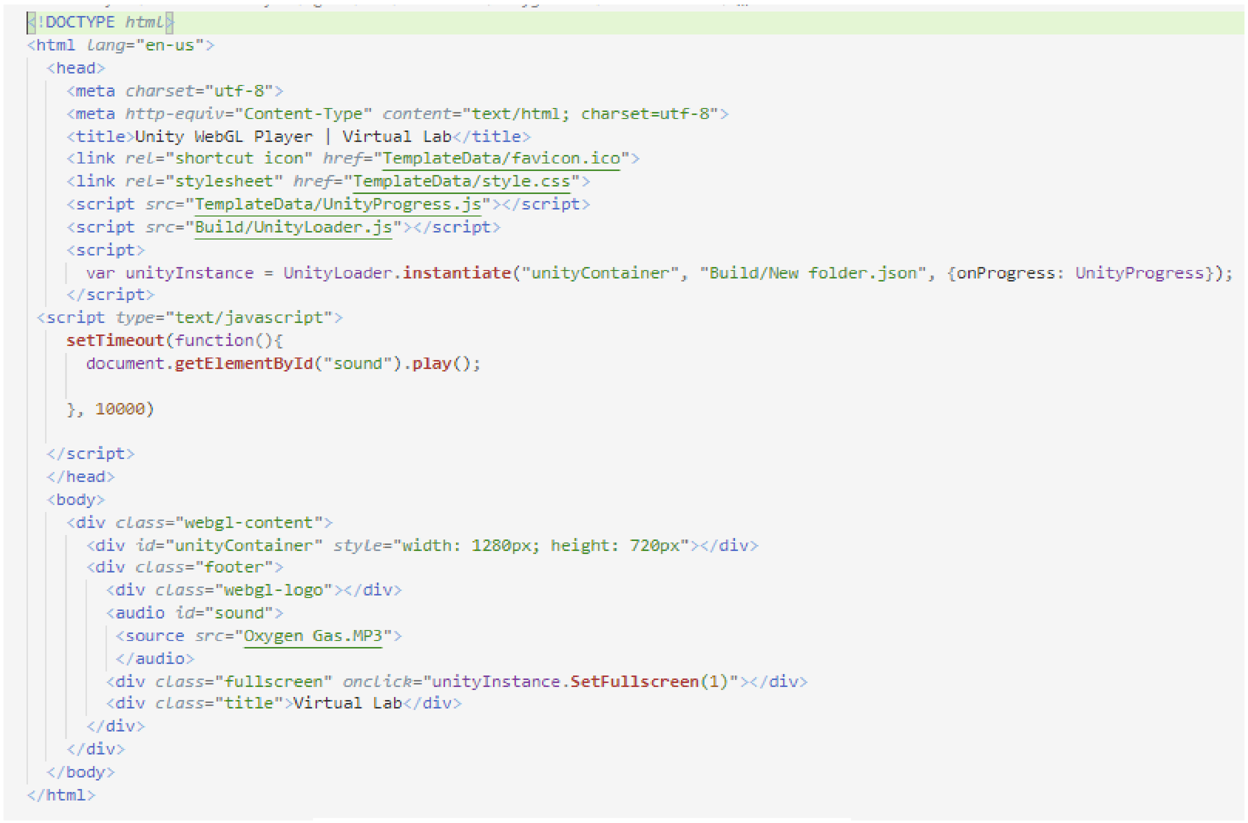
```html
<!DOCTYPE html>
<html lang="en-us">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Unity WebGL Player | Magnet</title>
    <link rel="shortcut icon" href="TemplateData/favicon.ico">
    <link rel="stylesheet" href="TemplateData/style.css">
    <script src="TemplateData/UnityProgress.js"></script>
    <script src="Build/UnityLoader.js"></script>
    <script>
      var unityInstance = UnityLoader.instantiate("unityContainer", "Build/Magnet.json", {onProgress: UnityProgress});
    </script>
    <script type="text/javascript">
      setTimeout(function(){
      document.getElementById("sound").play();

      }, 10000)

  </script>
  </head>
  <body>
    <div class="webgl-content">
      <div id="unityContainer" style="width: 960px; height: 600px"></div>
      <div class="footer">
        <div class="webgl-logo"></div>
         <audio id="sound">
           <source src="Magnet.MP3">
         </audio>
        <div class="fullscreen" onclick="unityInstance.SetFullscreen(1)"></div>
        <div class="title">Magnet</div>
      </div>
    </div>
  </body>
</html>
```

Figure 4.25: Html Magnet



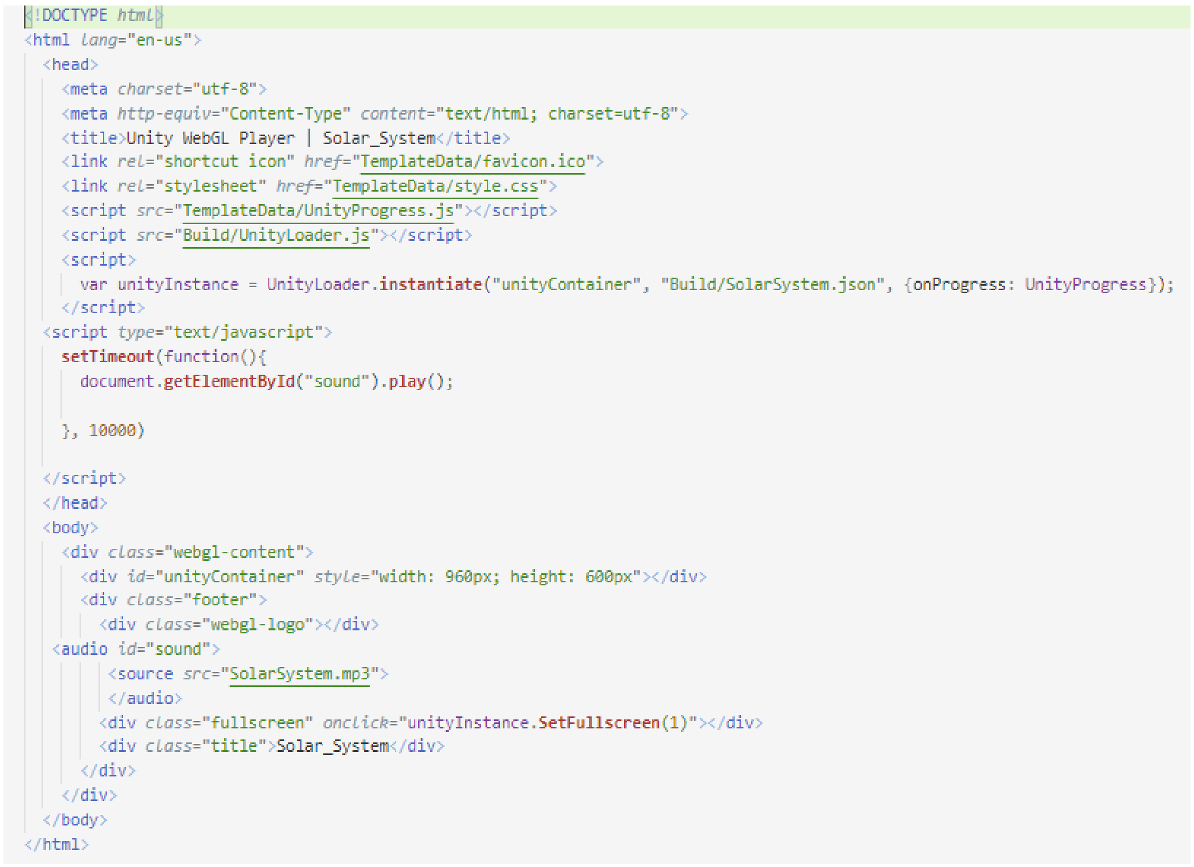
```html
<!DOCTYPE html>
<html lang="en-us">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Unity WebGL Player | Virtual Lab</title>
    <link rel="shortcut icon" href="TemplateData/favicon.ico">
    <link rel="stylesheet" href="TemplateData/style.css">
    <script src="TemplateData/UnityProgress.js"></script>
    <script src="Build/UnityLoader.js"></script>
    <script>
      var unityInstance = UnityLoader.instantiate("unityContainer", "Build/Carbon Dioxide Gas.json", {onProgress: UnityProgress});
    </script>
    <script type="text/javascript">
    setTimeout(function(){
      document.getElementById("sound").play();

    }, 10000)

  </script>
  </head>
  <body>
    <div class="webgl-content">
      <div id="unityContainer" style="width: 1280px; height: 720px"></div>
      <div class="footer">
        <div class="webgl-logo"></div>
         <audio id="sound">
         <source src="Carbon Dioxide Gas.MP3">
         </audio>
        <div class="fullscreen" onclick="unityInstance.SetFullscreen(1)"></div>
        <div class="title">Virtual Lab</div>
      </div>
    </div>
  </body>
</html>
```

Figure 4.26: Html Carbon Dioxide Gas



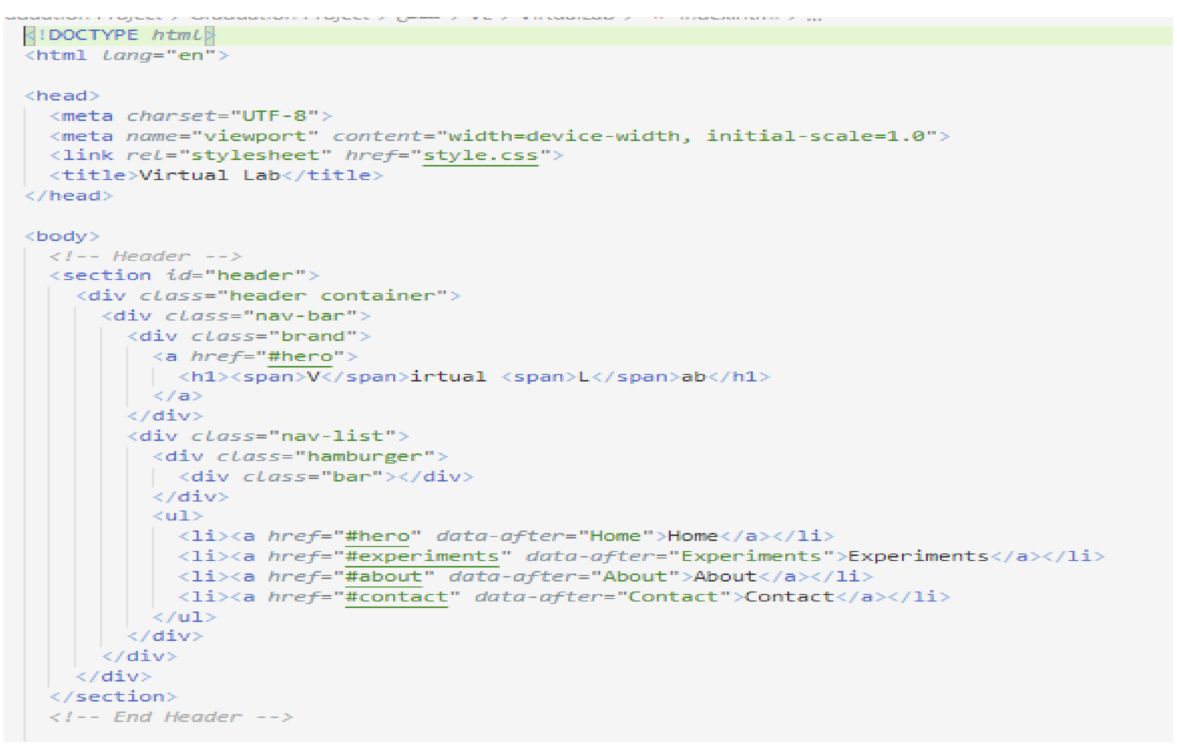
```html
<!DOCTYPE html>
<html lang="en-us">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Unity WebGL Player | Virtual Lab</title>
    <link rel="shortcut icon" href="TemplateData/favicon.ico">
    <link rel="stylesheet" href="TemplateData/style.css">
    <script src="TemplateData/UnityProgress.js"></script>
    <script src="Build/UnityLoader.js"></script>
    <script>
      var unityInstance = UnityLoader.instantiate("unityContainer", "Build/New folder.json", {onProgress: UnityProgress});
    </script>
 <script type="text/javascript">
    setTimeout(function(){
      document.getElementById("sound").play();

    }, 10000)

  </script>
  </head>
  <body>
    <div class="webgl-content">
      <div id="unityContainer" style="width: 1280px; height: 720px"></div>
      <div class="footer">
        <div class="webgl-logo"></div>
        <audio id="sound">
         <source src="Oxygen Gas.MP3">
         </audio>
        <div class="fullscreen" onclick="unityInstance.SetFullscreen(1)"></div>
        <div class="title">Virtual Lab</div>
      </div>
    </div>
  </body>
</html>
```

Figure 4.27: Html Oxygen Gas



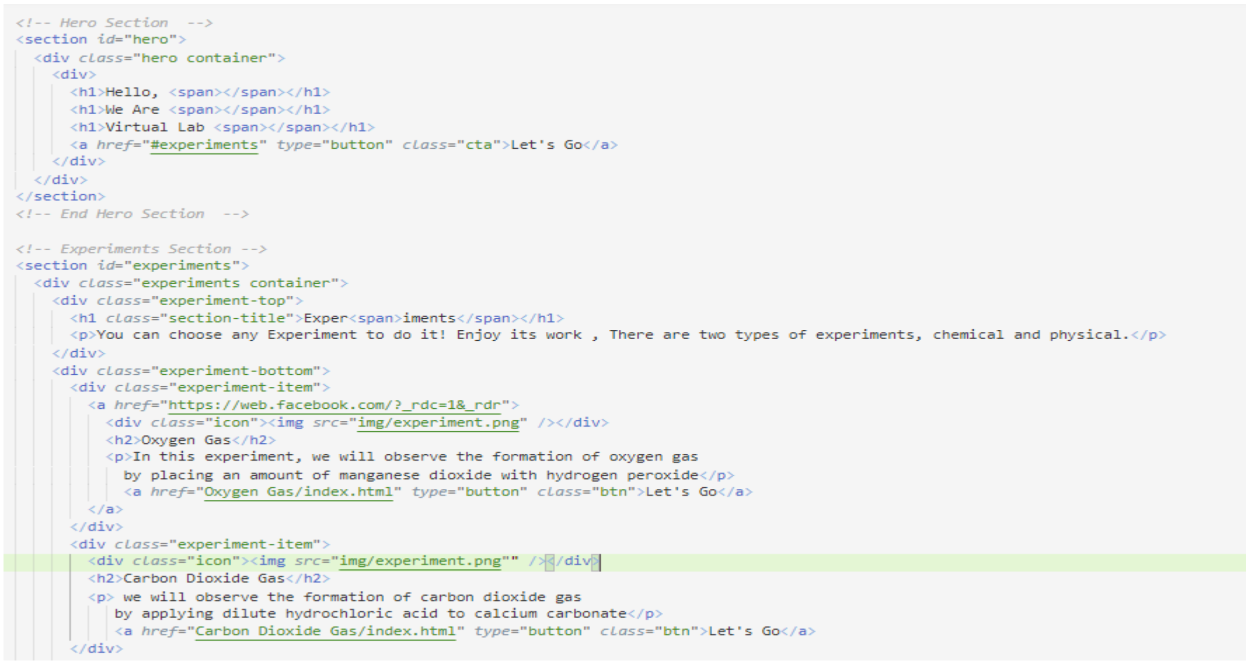
```html
<!DOCTYPE html>
<html lang="en-us">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Unity WebGL Player | Solar_System</title>
    <link rel="shortcut icon" href="TemplateData/favicon.ico">
    <link rel="stylesheet" href="TemplateData/style.css">
    <script src="TemplateData/UnityProgress.js"></script>
    <script src="Build/UnityLoader.js"></script>
    <script>
      var unityInstance = UnityLoader.instantiate("unityContainer", "Build/SolarSystem.json", {onProgress: UnityProgress});
    </script>
  <script type="text/javascript">
    setTimeout(function(){
      document.getElementById("sound").play();

    }, 10000)

  </script>
  </head>
  <body>
    <div class="webgl-content">
      <div id="unityContainer" style="width: 960px; height: 600px"></div>
      <div class="footer">
        <div class="webgl-logo"></div>
     <audio id="sound">
          <source src="SolarSystem.mp3">
          </audio>
        <div class="fullscreen" onclick="unityInstance.SetFullscreen(1)"></div>
        <div class="title">Solar_System</div>
      </div>
    </div>
  </body>
</html>
```

Figure 4.28: Html Solar System



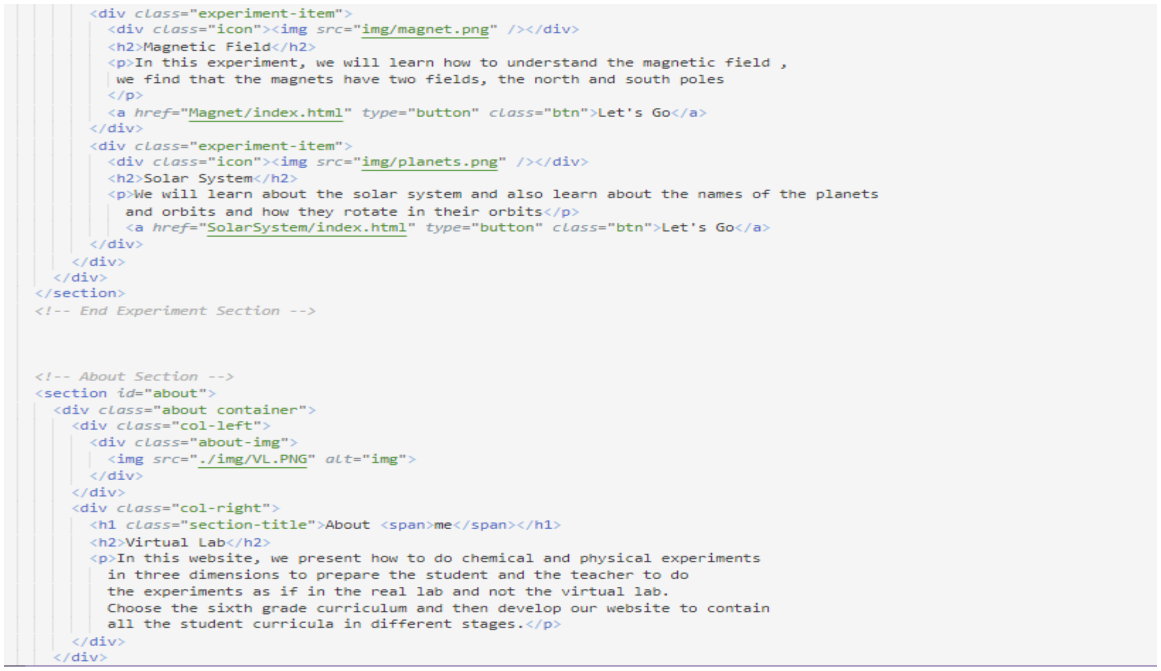
```html
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="style.css">
  <title>Virtual Lab</title>
</head>

<body>
  <!-- Header -->
  <section id="header">
    <div class="header container">
      <div class="nav-bar">
        <div class="brand">
          <a href="#hero">
            <h1><span>V</span>irtual <span>L</span>ab</h1>
          </a>
        </div>
        <div class="nav-list">
          <div class="hamburger">
            <div class="bar"></div>
          </div>
          <ul>
            <li><a href="#hero" data-after="Home">Home</a></li>
            <li><a href="#experiments" data-after="Experiments">Experiments</a></li>
            <li><a href="#about" data-after="About">About</a></li>
            <li><a href="#contact" data-after="Contact">Contact</a></li>
          </ul>
        </div>
      </div>
    </div>
  </section>
  <!-- End Header -->
```

Figure 4.29: Html Home Page part 1



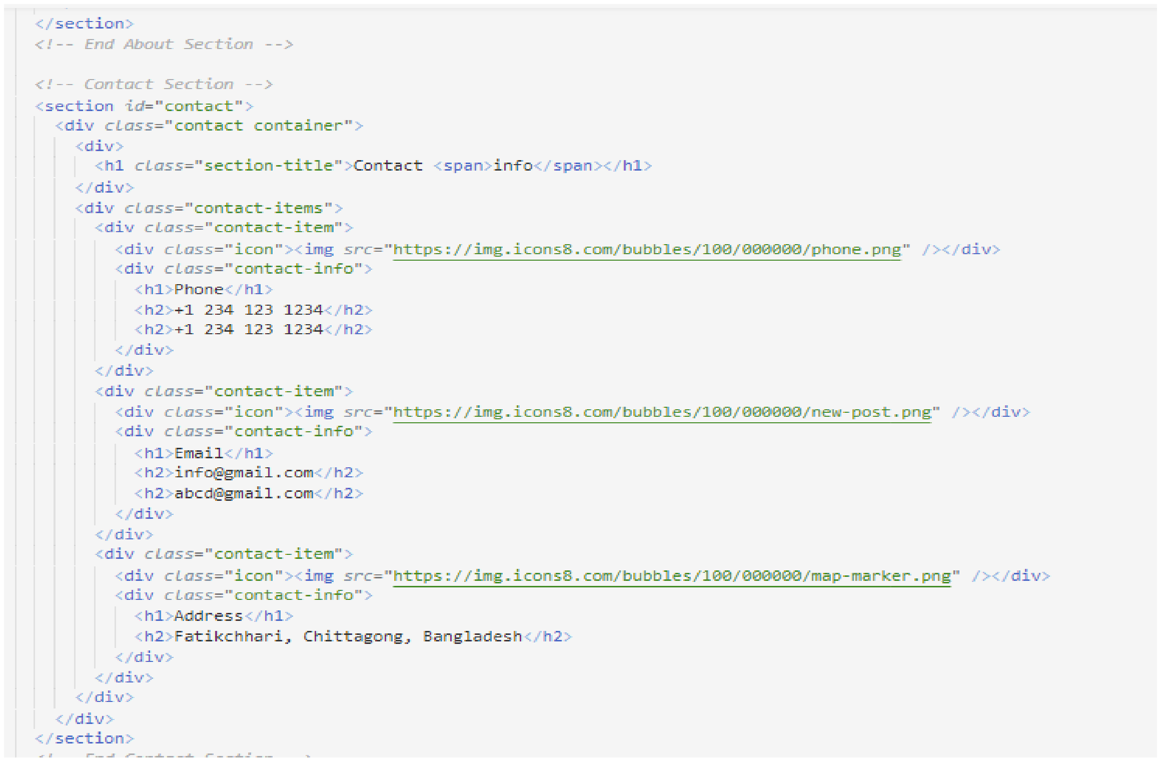
```html
<!-- Hero Section  -->
<section id="hero">
  <div class="hero container">
    <div>
      <h1>Hello, <span></span></h1>
      <h1>We Are <span></span></h1>
      <h1>Virtual Lab <span></span></h1>
      <a href="#experiments" type="button" class="cta">Let's Go</a>
    </div>
  </div>
</section>
<!-- End Hero Section  -->

<!-- Experiments Section -->
<section id="experiments">
  <div class="experiments container">
    <div class="experiment-top">
      <h1 class="section-title">Exper<span>iments</span></h1>
      <p>You can choose any Experiment to do it! Enjoy its work , There are two types of experiments, chemical and physical.</p>
    </div>
    <div class="experiment-bottom">
      <div class="experiment-item">
        <a href="https://web.facebook.com/?_rdc=1&_rdr">
          <div class="icon"><img src="img/experiment.png" /></div>
          <h2>Oxygen Gas</h2>
          <p>In this experiment, we will observe the formation of oxygen gas
            by placing an amount of manganese dioxide with hydrogen peroxide</p>
            <a href="Oxygen Gas/index.html" type="button" class="btn">Let's Go</a>
        </a>
      </div>
      <div class="experiment-item">
        <div class="icon"><img src="img/experiment.png"" /></div>
        <h2>Carbon Dioxide Gas</h2>
        <p> we will observe the formation of carbon dioxide gas
          by applying dilute hydrochloric acid to calcium carbonate</p>
          <a href="Carbon Dioxide Gas/index.html" type="button" class="btn">Let's Go</a>
      </div>
```

Figure 4.30: Html Home Page part 2



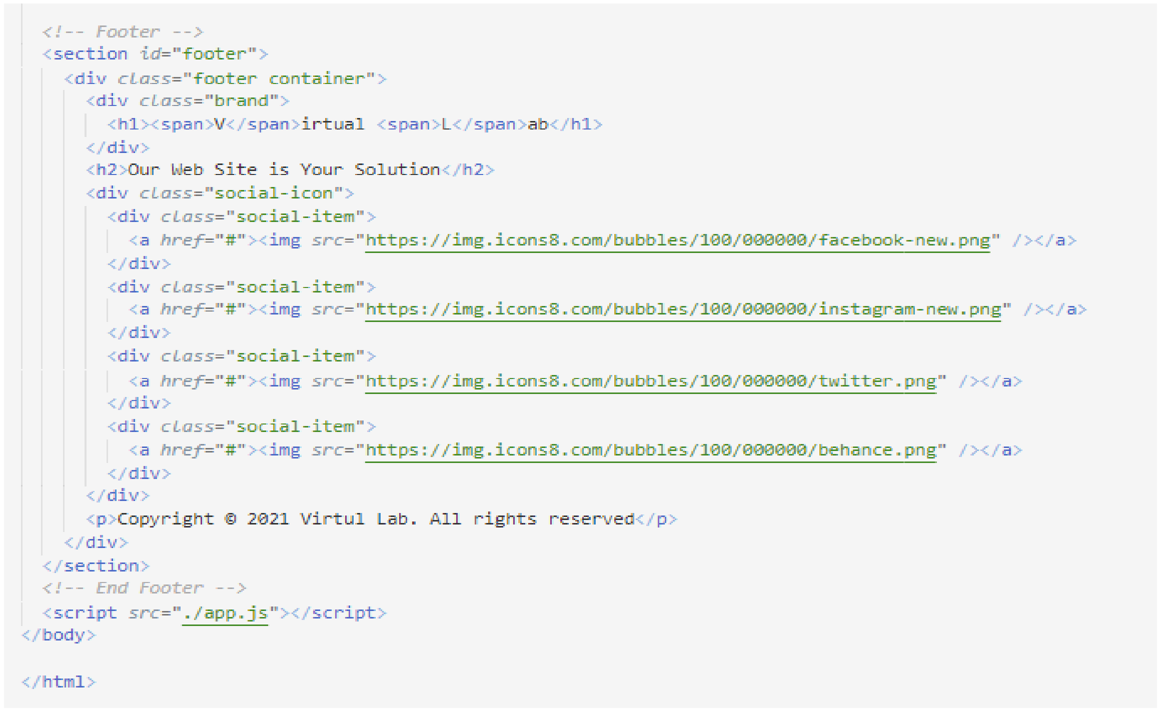
```html
        <div class="experiment-item">
          <div class="icon"><img src="img/magnet.png" /></div>
          <h2>Magnetic Field</h2>
          <p>In this experiment, we will learn how to understand the magnetic field ,
            we find that the magnets have two fields, the north and south poles
          </p>
          <a href="Magnet/index.html" type="button" class="btn">Let's Go</a>
        </div>
        <div class="experiment-item">
          <div class="icon"><img src="img/planets.png" /></div>
          <h2>Solar System</h2>
          <p>We will learn about the solar system and also learn about the names of the planets
            and orbits and how they rotate in their orbits</p>
            <a href="SolarSystem/index.html" type="button" class="btn">Let's Go</a>
        </div>
      </div>
    </div>
  </section>
  <!-- End Experiment Section -->


  <!-- About Section -->
  <section id="about">
    <div class="about container">
      <div class="col-left">
        <div class="about-img">
          <img src="./img/VL.PNG" alt="img">
        </div>
      </div>
      <div class="col-right">
        <h1 class="section-title">About <span>me</span></h1>
        <h2>Virtual Lab</h2>
        <p>In this website, we present how to do chemical and physical experiments
          in three dimensions to prepare the student and the teacher to do
          the experiments as if in the real lab and not the virtual lab.
          Choose the sixth grade curriculum and then develop our website to contain
          all the student curricula in different stages.</p>
      </div>
    </div>
```

Figure 4.31: Html Home Page part 3



```html
    </section>
    <!-- End About Section -->

    <!-- Contact Section -->
    <section id="contact">
      <div class="contact container">
        <div>
          <h1 class="section-title">Contact <span>info</span></h1>
        </div>
        <div class="contact-items">
          <div class="contact-item">
            <div class="icon"><img src="https://img.icons8.com/bubbles/100/000000/phone.png" /></div>
            <div class="contact-info">
              <h1>Phone</h1>
              <h2>+1 234 123 1234</h2>
              <h2>+1 234 123 1234</h2>
            </div>
          </div>
          <div class="contact-item">
            <div class="icon"><img src="https://img.icons8.com/bubbles/100/000000/new-post.png" /></div>
            <div class="contact-info">
              <h1>Email</h1>
              <h2>info@gmail.com</h2>
              <h2>abcd@gmail.com</h2>
            </div>
          </div>
          <div class="contact-item">
            <div class="icon"><img src="https://img.icons8.com/bubbles/100/000000/map-marker.png" /></div>
            <div class="contact-info">
              <h1>Address</h1>
              <h2>Fatikchhari, Chittagong, Bangladesh</h2>
            </div>
          </div>
        </div>
      </div>
    </section>
```

Figure 4.32: Html Home Page part 4



```html
    <!-- Footer -->
    <section id="footer">
      <div class="footer container">
        <div class="brand">
          <h1><span>V</span>irtual <span>L</span>ab</h1>
        </div>
        <h2>Our Web Site is Your Solution</h2>
        <div class="social-icon">
          <div class="social-item">
            <a href="#"><img src="https://img.icons8.com/bubbles/100/000000/facebook-new.png" /></a>
          </div>
          <div class="social-item">
            <a href="#"><img src="https://img.icons8.com/bubbles/100/000000/instagram-new.png" /></a>
          </div>
          <div class="social-item">
            <a href="#"><img src="https://img.icons8.com/bubbles/100/000000/twitter.png" /></a>
          </div>
          <div class="social-item">
            <a href="#"><img src="https://img.icons8.com/bubbles/100/000000/behance.png" /></a>
          </div>
        </div>
        <p>Copyright © 2021 Virtul Lab. All rights reserved</p>
      </div>
    </section>
    <!-- End Footer -->
    <script src="./app.js"></script>
  </body>

</html>
```

Figure 4.33: Html Home Page part 5

Arabic Summary of the project

كورونا اثرت على حياتنا كلنا بشكل ملحوظ و من اهم الاشياء التى تأثرت بسبب انتشار كورونا هى المنظومة التعليمية لذلك نقرر نعمل موقع نساعد بيهم الطلاب و المعلمين بانهم ينفذون التجارب العملية بدون اى ضرر او خسائر و ليس فقط السبب الوحيد هو كورونا و لكن يوجد اسباب اخرى مثل ان اسعار الادوات الكيميائية غالية الثمن و يوجد بعض المدارس الحكومية التى لاتتوفر بها الادوات بسبب ثمنها الغالى و ايضا يوجد بعض الاهمال من المعلم اثناء وجوده فى المعمل مع الطلاب لذلك لعدم تعرض الطالب لاى مخاطر داخل المعمل قمنا بعمل موقع يحاكى المعمل و يسمى المعمل الافتراضى حيث يقوم الطالب او المعلم بعمل التجربة دون مخاطر و يقوم الطالب بعمل التجربة فى اى وقت ليس فقط فى المعمل المدرسي كما ان الموقع يتح للطالب المعمل الافتراضى بشكل ثلاثى الابعاد كما يجعل الطالب كأنه فى المعمل المدرسي كما ان يوفر على المدرسة شراء هذه الادوات من خلال استخدام هذا الموقع .

سوف يقوم الطالب بالدخول الى الموقع ثم بعد ذلك يقوم باختيار التجربة المراد عملها و قبل البدء فى التجربة يقوم ملف صوتى فى بداية كل تجربة لتوضح كيفية عمل التجربة بسهوله و يسر دون اى اخطاء يقع فيها الطالب ثم بعد ذلك يقوم بعمل تجربته وبعد الانتهاء من التجربة يرأى كل النتائج كما انه معمله المدرسى .

الموقع مصمم بشكل جاذبي و ايضا سهل الاستخدام والتعامل معه.