

CS 550 | Fall 2023

Project 4

“Keytime Animation”

By

Pushpak Katkhede

Email – katkhedp@oregonstate.edu

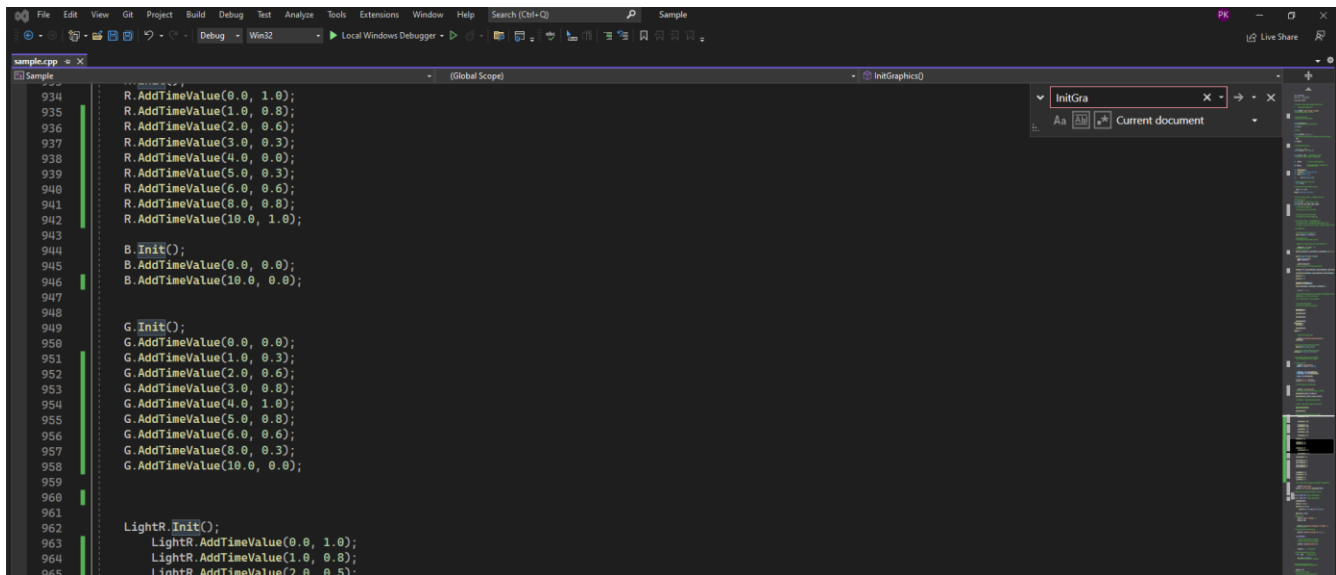
OUSID – 934453040

Instructor – Prof Mike Bailey

❖ Description →

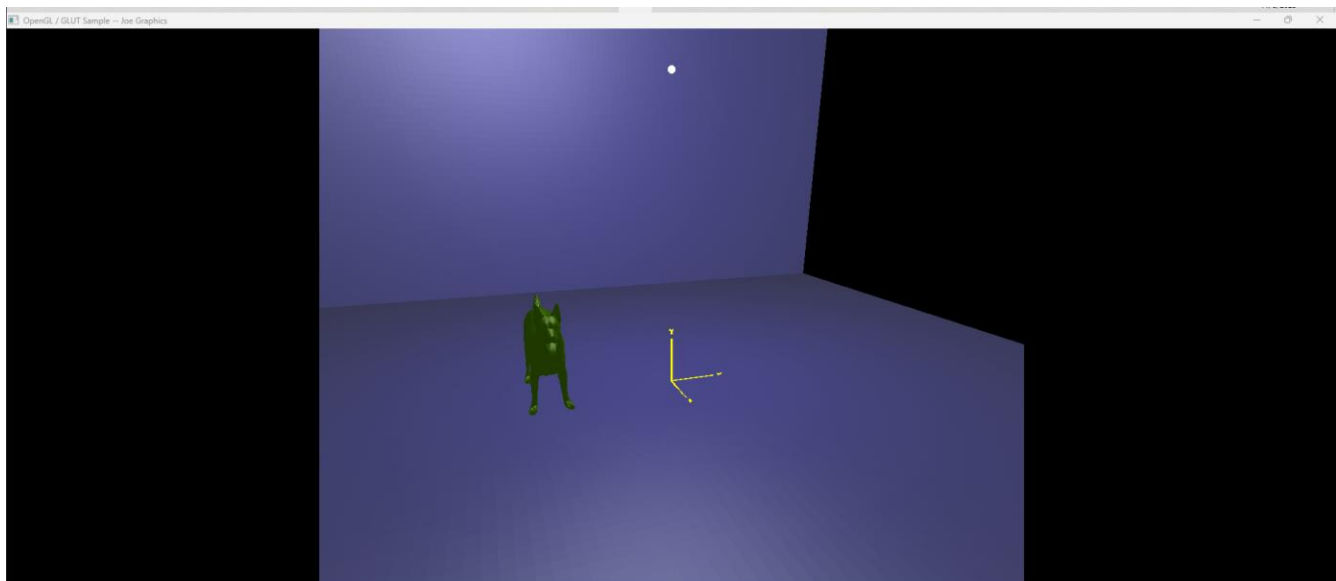
Video Link: [Click Here](#)

❖ Screenshots →



The screenshot shows the Visual Studio Code editor with a C++ file named `sample.cpp`. The code is organized into several functions: `R.AddTimeValue`, `B.Init`, `G.Init`, and `LightR.Init`. The `R.AddTimeValue` function is called multiple times with different parameters. The `B.Init` function calls `B.AddTimeValue`. The `G.Init` function calls `G.AddTimeValue`. The `LightR.Init` function calls `LightR.AddTimeValue`. The code is written in a dark theme. The `sample.cpp` file is open in the editor, and the `Global Scope` and `InitGraphics()` are visible in the sidebar. The `sample.cpp` file is open in the editor, and the `Global Scope` and `InitGraphics()` are visible in the sidebar.

```
934 R.AddTimeValue(0.0, 1.0);
935 R.AddTimeValue(1.0, 0.8);
936 R.AddTimeValue(2.0, 0.6);
937 R.AddTimeValue(3.0, 0.3);
938 R.AddTimeValue(4.0, 0.0);
939 R.AddTimeValue(5.0, 0.3);
940 R.AddTimeValue(6.0, 0.6);
941 R.AddTimeValue(8.0, 0.8);
942 R.AddTimeValue(10.0, 1.0);
943
944 B.Init();
945 B.AddTimeValue(0.0, 0.0);
946 B.AddTimeValue(10.0, 0.0);
947
948 G.Init();
949 G.AddTimeValue(0.0, 0.0);
950 G.AddTimeValue(1.0, 0.3);
951 G.AddTimeValue(2.0, 0.6);
952 G.AddTimeValue(3.0, 0.8);
953 G.AddTimeValue(4.0, 1.0);
954 G.AddTimeValue(5.0, 0.8);
955 G.AddTimeValue(6.0, 0.6);
956 G.AddTimeValue(8.0, 0.3);
957 G.AddTimeValue(10.0, 0.0);
958
959
960
961 LightR.Init();
962 LightR.AddTimeValue(0.0, 1.0);
963 LightR.AddTimeValue(1.0, 0.8);
964 LightR.AddTimeValue(2.0, 0.5);
965
```



```
#object position
Xpos.Init();
    Xpos.AddTimeValue(0.0, 0.000);
    Xpos.AddTimeValue(1.0, 3.0);
    Xpos.AddTimeValue(2.0, 6.0);
    Xpos.AddTimeValue(3.0, 3.0);
    Xpos.AddTimeValue(4.0, 0.0);
    Xpos.AddTimeValue(5.0, -3.0);
    Xpos.AddTimeValue(6.0, -6.0);
    Xpos.AddTimeValue(8.0, -3.0);
    Xpos.AddTimeValue(10.0, 0.0);
```

```
Ypos.Init();
    Ypos.AddTimeValue(0.0, 0.000);
    Ypos.AddTimeValue(1.0, 1);
    Ypos.AddTimeValue(2.0, 0.0);
    Ypos.AddTimeValue(3.0, 2);
    Ypos.AddTimeValue(4.0, 0.0);
    Ypos.AddTimeValue(5.0, 1);
    Ypos.AddTimeValue(6.0, 0.0);
    Ypos.AddTimeValue(7.0, 2);
    Ypos.AddTimeValue(8.0, 0.0);
    Ypos.AddTimeValue(9.0, 1);
    Ypos.AddTimeValue(10.0, 0.0);
```

```
Zpos.Init();
    Zpos.AddTimeValue(0.0, 0.000);
    Zpos.AddTimeValue(1.0, 2.0);
    Zpos.AddTimeValue(2.0, 3.0);
    Zpos.AddTimeValue(3.0, 4.0);
    Zpos.AddTimeValue(4.0, 3.0);
    Zpos.AddTimeValue(5.0, 2.0);
    Zpos.AddTimeValue(6.0, 0.0);
    Zpos.AddTimeValue(7.0, -3.0);
    Zpos.AddTimeValue(8.0, -6.0);
    Zpos.AddTimeValue(9.0, -3.0);
    Zpos.AddTimeValue(10.0, 0.0);
```

```
#Object roational axis
XRot.Init();
    XRot.AddTimeValue(0.0, 0.000);
    XRot.AddTimeValue(1.0, 10);
    XRot.AddTimeValue(2.0, 0.0);
    XRot.AddTimeValue(3.0, 20);
    XRot.AddTimeValue(4.0, 0.0);
    XRot.AddTimeValue(5.0, 10);
    XRot.AddTimeValue(6.0, 0.0);
    XRot.AddTimeValue(7.0, 20);
    XRot.AddTimeValue(8.0, 0.0);
    XRot.AddTimeValue(9.0, 10);
    XRot.AddTimeValue(10.0, 0.0);
```

```
#object coloring
R.Init();
    R.AddTimeValue(0.0, 1.0);
    R.AddTimeValue(1.0, 0.8);
    R.AddTimeValue(2.0, 0.6);
    R.AddTimeValue(3.0, 0.3);
```

```
R.AddTimeValue(4.0, 0.0);
R.AddTimeValue(5.0, 0.3);
R.AddTimeValue(6.0, 0.6);
R.AddTimeValue(8.0, 0.8);
R.AddTimeValue(10.0, 1.0);
```

```
G.Init();
G.AddTimeValue(0.0, 0.0);
G.AddTimeValue(1.0, 0.3);
G.AddTimeValue(2.0, 0.6);
G.AddTimeValue(3.0, 0.8);
G.AddTimeValue(4.0, 1.0);
G.AddTimeValue(5.0, 0.8);
G.AddTimeValue(6.0, 0.6);
G.AddTimeValue(8.0, 0.3);
G.AddTimeValue(10.0, 0.0);
```

#lighting Colour

```
LightR.Init();
LightR.AddTimeValue(0.0, 1.0);
LightR.AddTimeValue(1.0, 0.8);
LightR.AddTimeValue(2.0, 0.5);
LightR.AddTimeValue(3.0, 0.2);
LightR.AddTimeValue(4.0, 0.0);
LightR.AddTimeValue(5.0, 0.2);
LightR.AddTimeValue(6.0, 0.5);
LightR.AddTimeValue(7.0, 0.8);
LightR.AddTimeValue(8.0, 0.0);
LightR.AddTimeValue(9.0, 0.8);
LightR.AddTimeValue(10.0, 1.0);
```

```
LightG.Init();
LightG.AddTimeValue(0.0, 1.0);
LightG.AddTimeValue(1.0, 0.8);
LightG.AddTimeValue(2.0, 0.5);
LightG.AddTimeValue(3.0, 0.2);
LightG.AddTimeValue(4.0, 0.0);
LightG.AddTimeValue(5.0, 0.2);
LightG.AddTimeValue(6.0, 0.5);
LightG.AddTimeValue(7.0, 0.8);
LightG.AddTimeValue(8.0, 0.0);
LightR.AddTimeValue(9.0, 0.8);
LightG.AddTimeValue(10.0, 1.0);
```

#lighting Source Position

```
LightPosy.Init();
LightPosy.AddTimeValue(0, 5);
LightPosy.AddTimeValue(2, 6);
LightPosy.AddTimeValue(4, 7);
LightPosy.AddTimeValue(6, 10);
LightPosy.AddTimeValue(8, 12);
LightPosy.AddTimeValue(10, 15);
```

#eye-view location

```
Xeye.Init();
Xeye.AddTimeValue(0, 15);
Xeye.AddTimeValue(1, 7.5);
Xeye.AddTimeValue(2, 0);
Xeye.AddTimeValue(3, -7.5);
Xeye.AddTimeValue(4, -15);
```

```

Xeye.AddTimeValue(5, -7.5);
Xeye.AddTimeValue(6, 0);
Xeye.AddTimeValue(8, 7.5);
Xeye.AddTimeValue(10, 15);

Yeye.Init();
Yeye.AddTimeValue(0, 8.0);
Yeye.AddTimeValue(1.0, 6.5);
Yeye.AddTimeValue(2.0, 4.0);
Yeye.AddTimeValue(3.0, 3.0);
Yeye.AddTimeValue(4.0, 4.5);
Yeye.AddTimeValue(5.0, 6.0);
Yeye.AddTimeValue(6.0, 7.50);
Yeye.AddTimeValue(8.0, 8.0);

Zeye.Init();
Zeye.AddTimeValue(0, 25);
Zeye.AddTimeValue(1, 22.5);
Zeye.AddTimeValue(2, 20);
Zeye.AddTimeValue(3, 17.5);
Zeye.AddTimeValue(4, 15);
Zeye.AddTimeValue(6, 15);
Zeye.AddTimeValue(7, 17.5);
Zeye.AddTimeValue(8, 20);
Zeye.AddTimeValue(9, 22.5);
Zeye.AddTimeValue(10, 25);

```

Description:

In the given code, I've set up various parameters for an animation or scene that involves a dog playing outside in an open space. Let me break down these elements:

1. **Object Position:** I've specified how the dog's position changes over time. The dog moves in the X, Y, and Z directions, creating a dynamic and lifelike animation. It starts at the origin (0, 0, 0) and follows a predefined path.
2. **Object Rotational Axis:** I've defined the dog's rotation around the X-axis. This allows the dog to perform actions like jumping, spinning, or other rotational movements. The rotation angle varies over time to create dynamic motion.
3. **Object Coloring:** I've made the dog's color change gradually as it moves through the scene. The RGB values for the dog's color are defined at different time points, creating a smooth transition from one color to another.
4. **Lighting Color:** I've adjusted the lighting in the scene to change over time, affecting the overall ambiance and mood. This adjustment involves the red and green components of the lighting color, creating a dynamic lighting effect.
5. **Lighting Source Position:** I've made the light source's position change with time, impacting how the scene is illuminated. This adds interesting shadows and highlights to the animation.
6. **Eye-View Location:** I've also changed the viewpoint or perspective from which I observe the scene over time. This adds a dynamic aspect to the animation, allowing me to experience different angles and views of the dog's playful activities.

Altogether, these parameters work in unison to create a dynamic and visually engaging animation of a dog enjoying outdoor playtime. This results in changing positions, colors, lighting, and viewpoints, making the scene more captivating and immersive.

Based on the above information and visualized output I'm confident in the animation's success for several reasons. It effectively conveys dynamic dog movement in three dimensions, accurately represents rotational actions, smoothly transitions colors, and skillfully manages dynamic lighting to enhance the scene's mood. Furthermore, it offers diverse viewpoints, ensuring a consistently engaging and immersive experience. These elements combine harmoniously to achieve the animation's objective of showcasing a playful dog in an outdoor setting.