Producing Animated Sinusoid Model

Nathan Ulmer

Professor: Dr. Jeyaprakash Chelladurai

COMP430: Interactive Computer Graphics

Project Report

Department of Applied Computer Science

Lock Haven University of Pennsylvania

Lock Haven, PA, USA

Fall 2018

1. Introduction

1.1 Background

A sinusoid is a function that models periodic oscillation. Sine waves can be used to model all sorts of occurrences in the natural world such as the action of a pendulum or the movement of sound waves. A sine function has several components: amplitude, frequency, angular frequency, and phase. Utilizing the function itself, a script to resize objects within a 3D modeling environment along with the generation of keyframes over a specified time interval can be used to create an animation of a sine wave.

1.2 Creating Sine Wave Animation

Periodic oscillation can be mathematically described as a function that smoothly increases and decreases in a repeating pattern over time. To model this, a series of objects height can be manipulated over time. By setting each objects height to a position along a sine wave, the initial wave is established. A key frame is inserted to save this initial wave at time 0. The period of the sine wave is then adjusted by one position at the size of one object. Each object's height along the sine wave is then adjusted accordingly, and then another keyframe is inserted to save the second position of the wave. This process is repeated for the desired duration of the animation.

1.3 Contribution

The purpose of this project is to create a sine wave visualization that is fully customizable. The project will utilize blender python scripting, otherwise known as the bpy library. The bpy library can fully automate and control all functions that can be accessed through the interface, and in some cases, with a greater deal of control.

1.4 Organization

This report is organized into 4 sections. Section 1 gives a brief description of the project. Section 2 will give a formal description of the major components of the project. Section 3 will describe the implementation of the techniques used to create the project. Section 4 will be a conclusion of the project and will include possible modifications as well as other projects that could build upon the foundation laid by this one.

2 Formal Description

To create the animation and ensure that it is adaptable to different scales, it is necessary to convert the abstract parameter to real-world coordinates. For example, an objects location within a loop will be defined by an integer value, to adapt this to the scale that has been set, the abstract integer value will be converted to a percent. The lower bound of the real-world coordinates will be added to this number, and then the sum multiplied by the upper bound of the coordinates subtracted from the lower bounds. Each object then will be scaled along the z axis for every key frame created. Using the sine function and nested for loops, each object will be assigned a height at every interval. When the final animation is created, blender will interpolate the change in height between each key frame, resulting in a smooth animated wave.

3 Implementation

In this section, the implementation of the methods used to create the sine wave will be discussed. Excerpts from the blender script will be displayed and described in further detail.

3.1 Abstract to Real-World Coordinates

```
count = 100

scale = 10.0

padding = 0.002

size = (extents / count) – padding

min_size = size * 0.25

max_size = size * extents * 10

size_difference = max_size – min_size
```

Initializing the above values sets the number of objects (count) and their abstract scale that will demonstrate the sine wave. Since the height of the objects will be animated, a minimum size and a maximum size is required, which will ultimately be dependent on the extent of the model.

3.2 Sine Wave and Key Frame Generation

A nested for loop is used to create and set the scale of each object for every key frame. The first for loop creates the object and calculates its position, as well as the rise and run for the initial height. The k and j percent determine the scale of the object based off of the abstract scale initially set.

The normalized distance will also be used to create a HSV (Hue Saturation Value) that will color the object.

```
for j in range(0, count, 1):
    k_percent = j * count_fraction
    x = -scale + k_percent * scale_difference

    j_percent = j * count_fraction
    y = -scale + j_percent * scale_difference

    rise = y - center_y
    rise *= rise

    run = x - center_x
    run *= run
    normalized_distance = sqrt(rise + run) / max_distance
    off_set = -TWOPI * normalized_distance + pi

     bpy.ops.mesh.primitive_ico_sphere_add(location=(center
     _x + x, center_y, center_z), size=size)
```

The second for loop calculates the scale of the object along the z axis for each keyframe. First the current frame is set to initial key frame, the scale of the object in the current key frame is calculated. To change the scale along the z axis, the object's scale matrix is adjusted at index 2. The current frame is then incremented.

```
current_frame = bpy.context.scene.frame_start

for f in range(0, keyframe_count, 1):
    f_percent = f * inverted_frame_count
    angle = TWOPI * f_percent
    bpy.context.scene.frame_set(current_frame)

    current.scale[2] = min_size + abs(sin(off_set +
    angle)) * size_difference

    current.keyframe_insert(data_path='scale', index=2)
    current_frame += frame_increment
```

3.3 Hue Saturation Value

To make the model more visually appealing, HSVs can be generated from the normalized distance value. The offset used to calculate the normalized difference sets the initial height of each object, creating the wave as each object's height is rescaled in each frame. This offset also allows for the color of each object to be based off its initial height, giving each a differentiated hue.

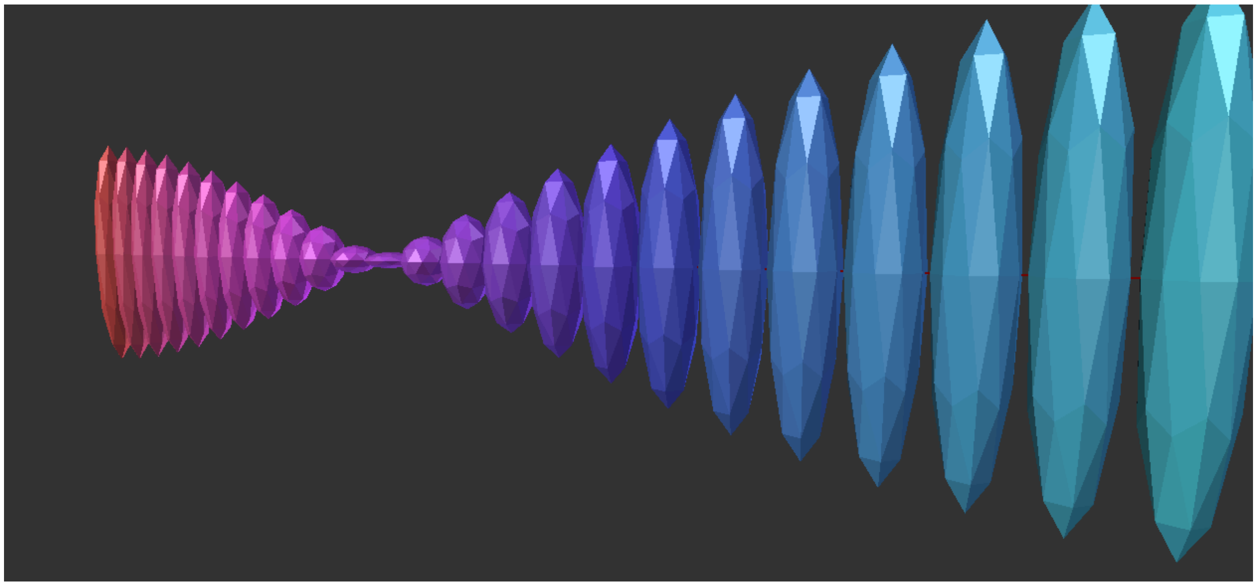
```
    material =
    bpy.data.materials.new(name='Material({0:0>2d})'.format(j))

    material.diffuse_color =
    colorsys.hsv_to_rgb(normalized_distance, 1.0, 1.0)

    current.data.materials.append(material)
```

A new material is created with the diffuse color option. The colorsys library is then used to convert a HSV to a RGB. HSV is a color based on its hue and saturation.

4 Conclusions

4.1 Results

The sine wave model can be seen in figure 1. If the model within the blender application is animated and can be looped to give a smooth wave animation. Each object will change in size as the animation plays.



Figures 1 Sine Wave Model

4.2 Future Work

Utilizing the same process for generating the sine wave, other mechanical motions can be modeled and animated as well.

The formula for projectile motion could be easily converted into a bpy script. Setting keyframes and object position along the curve of motion, interpolation could be used to project the object along its path. Modifying the interpolation, a more realistic animation could be achieved to model the objects acceleration.