Programme Code: TU857, TU856, TU856, TU984
Module Code: CMPU 4030
CRN: 22528, 22418, 31083, 27945

TECHNOLOGICAL UNIVERSITY DUBLIN

KEVIN STREET CAMPUS

BSc. (Honours) Degree in Computer Science (Infrastructure)

BSc. (Honours) Degree in Computer Science

BSc. (Honours) Degree in Computer Science (International)

Year 4

BA. (Honours) in Game Design

Year 3

SEMESTER 1 EXAMINATIONS 2020/21

Games Engines 1

Dr. Bryan Duggan
Dr. Deirdre Lillis
Ms. Pauline Martin
Ms. Sanita Tifentale

TIME AND DATE: TBC

Question 1

A 3D tower defence game made in Unity has the following rules:

1. There is a crosshair in the centre of the screen that the player uses to select a location on the map in order to place a tower.
2. Towers become active when a creep comes in range.
3. When a tower becomes active it will turn to face the creep and continue targeting the creep so long as it stays in range.
4. Towers can fire 5 bullets per second.
5. Bullets disappear after 5 seconds if they don't hit anything
6. There are three possible types of creeps and each has an equal probability of being spawned.
7. Creeps follow a path to get to the players base.
8. When a creep is hit with a bullet, it explodes and after a few seconds, sinks into the ground and gets removed from the scene.

Taking each of the rules above, how you would program them in Unity?

(8 x 5 marks)

Solution:

1. Generate a picking ray from the camera:
Ray ray = camera.ScreenPointToRay(Input.mousePosition);
Project the ray into the scene with a layer mask for the ground.
If the RayCast returns true, Get the position from the RayCastHitInfo output parameter

2. Add a SphereCollider to the tower game object. Set the isTrigger to be true. Add a game component that has the OnTriggerEnter Method. Add a bool flag called firing and set it to be true

3. Add the OnTriggerStay method to the tower gamecomponent. Calculate the rotation required to face the player by using Quaternion.LookRotation. Slerp the current rotation to the calculated rotation. You could add a turn speed parameter

4. Use a coroutine. Start the coroutine in OnEnable. Inside the coroutine should be an infinite loop that checks the firing flag and if true, fires a bullet by instiantiating a bullet game object from a bullet prefab. Set the bullet position and rotation. Then sleep for 1/5 seconds

5. Use Invoke with a time parameter to invoke a method after 5 seconds. The method should destroy the bullet gameobject

6. Make a public field of type array of GameObject. Assign 3 elements of the array to be the three creep prefabs. Generate a random number between 0 and 3. Use this to index into the array. Use GameObject.Instiantiate to create the creep

7. Create a public List of Vector3 containing the waypoints. You could populate this automatically from gameobjects. Make an int variable for the current waypoint and a float variable for speed. In Update you could use Vector3.MoveTowards(transform.position, waypoints[current], speed * Time.deltaTime). Check the distance < some small amount and if so, waypoints++

8. If the creep was made from different gameobjects, you could add rigid bodies to each of them and then use AddExplosionForce to each one. Use a coroutine that first delays for a few second to allow the gameobject to come to rest then removes the rigidbody or sets it to be kinematic. The coroutine should then translate the gameobject on the Y axis for a few seconds and then callGameObject.Destroy on the gameobjct to remove it from the scene

Question 2

(a) Discuss the relationship between the quantities of *force*, *velocity*, *acceleration*, *position*, *distance*, *time* and *mass* in relation to 3D computer games. In your answer include:

i. Units of measurement and representations for these quantities. (5 marks)
ii. Equations that describe the relationships. (5 marks)
iii. A description of how to update the state of a Newtonian physics particle with respect to time in a 3D computer game. (5 marks)

Solution:

i.

Symbol	Value	Type	Units	Description
f	Force	Vector	Newtons	Alters an objects speed by affecting acceleration
p	Position	Vector	X, Y Z in meters	
a	Acceleration	Vector	M/S/S	Rate in change in velocity
v	Velocity	Vector	M/S	Speed in a direction. Magnitude gives the speed.
t	Time	Scalar	Seconds	
m	Mass	Scalar	Kg	

(5 marks)

ii.

$v = \Delta x / \Delta t$ - Change in distance / Change in time
$a = \Delta v / \Delta t$ – Change in velocity / Change in time
$\Delta v = a \Delta t$ – Change in velocity = acceleration * time
$\Delta x = u \Delta t + .5 a \Delta t^2$ - Distance = start velocity * change in time + .5 acceleration * change in time squared
$f = m/a$
$a = f / m$ – Acceleration = Force / Mass

(5 marks)

iii.

Assume the particle keeps track of Mass, position, velocity, we need to update these given a time delta:

```
A = F / M
```

V = V + A T

X = X + V T

(5 marks)

(b) A *gravity gun* in 3D games allows the player to grab an object and hold it at a point in front of the camera. Explain in detail how you would implement a gravity gun effect in a Unity project.

(15 marks)

```
if (Input.GetMouseButton(0))
        {
            if (pickedUp == null)
            {
                RaycastHit raycastHit;

                if (Physics.Raycast(transform.position, transform.forward, out raycastHit))
                {
(3 marks)
                    if (raycastHit.collider.gameObject.tag != "groundPlane")
                    {
(1 mark)
                        pickedUp = raycastHit.collider.gameObject;
                    }
                    else
                    {
                        pickedUp = null;
                    }
                }
            }
            else
            {
                Vector3 holdPos = transform.position + (transform.forward * holdDistance);
(2 marks)
                //LineDrawer.DrawTarget(holdPos, Color.yellow);
                // Move the object towards the holdPos
                Vector3 toHoldPos = holdPos - pickedUp.transform.position;
                toHoldPos *= powerfactor;
                toHoldPos = Vector3.ClampMagnitude(toHoldPos, maxVelocity);
(4 marks)
                // Alternatively...
               pickedUp.GetComponent<Rigidbody>().velocity = toHoldPos;
(2 marks)
            }
        }
        else
        {
            pickedUp = null;
        }
      }
```

Question 3

(a) **Error! Reference source not found.** shows an extract from a generative physics system that creates the caterpillar animal given in Figure 2.

```
 void Awake()
{
    float depth = size * 0.05f;
    Vector3 start = - Vector3.forward * bodySegments * depth * 2;
    GameObject previous = null;
    for (int i = 0; i < bodySegments; i++)
    {
        float mass = 1.0f;
        GameObject segment = GameObject.CreatePrimitive(PrimitiveType.Cube);
        Rigidbody rb = segment.AddComponent<Rigidbody>();
        rb.useGravity = gravity;
        rb.mass = mass;
        segment.name = "segment " + i;
        Vector3 pos = start + (Vector3.forward * depth * 4 * i);
        segment.transform.position = transform.TransformPoint(pos);
        segment.transform.rotation = transform.rotation;
        segment.transform.parent = this.transform;
        segment.transform.localScale = new Vector3(size, size, depth);
        segment.GetComponent<Renderer>().shadowCastingMode =
UnityEngine.Rendering.ShadowCastingMode.Off;
        segment.GetComponent<Renderer>().receiveShadows = false;

        segment.GetComponent<Renderer>().material.color = Color.HSVToRGB(i /
(float)bodySegments, 1, 1);

        if (previous != null)
        {
            j.autoConfigureConnectedAnchor = false;
            j.anchor = new Vector3(0, 0, -2f);
            j.connectedAnchor = new Vector3(0, 0, 2f);
            j.axis = Vector3.right;
            j.useSpring = true;
            JointSpring js = j.spring;
            js.spring = spring;
            js.damper = damper;
            j.spring = js;              }
        previous = segment;
    }
}
```

Figure 1

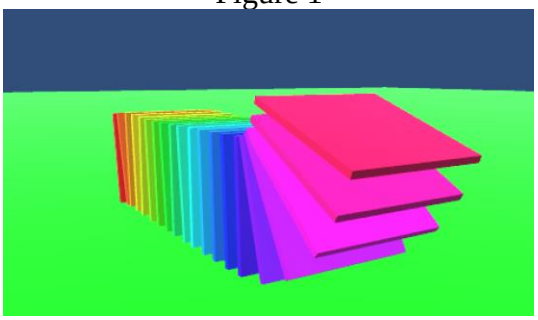
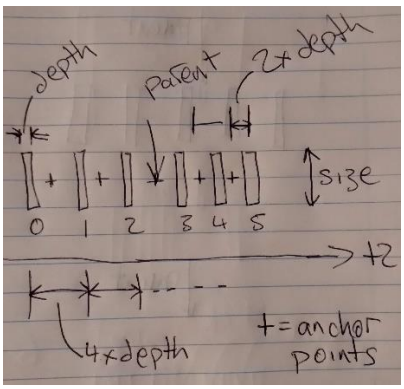


Figure 2

Answer these questions about the code:

(a) How are the position, rotation, anchor points and scale of each segment calculated? Include a diagram in your answer



(5 marks)

Depth is calculated as 5% of the size
The segments are scaled to size, size and depth so that that are like tiles rotated 90 deg
Position calculations are done in local space
Segments are laid out from -z to +z
There is a gap of 4x the depth between each segment
The segments are centred around the transform
The local space segment positions are transformed to world space using the transform
The segments are rotated to the same rotation as the transform
The anchor points are situated at the point between the two segments. 2 units behind and in front

(5 marks)

(b) How are the segments constrained to move relative to one another?

(10 marks)

Hinge – allows them to rotate relative to each other. They can rotate around the right axis (see diagram). The anchor point is 2 units behind the segment. This is the middle point between the segments as they are 4 units apart. The connected anchor is the point on the connected rigidbody. This is 2 units in front of the connected rigid body.

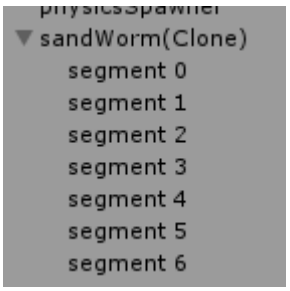
They are also connected via a spring that will try and move the segments back to their original positions and rotations relative to each other. When torque is applied they will spring back

(c) How is the colour of each segment determined?

(3 marks)

The colour is procedurally generated by stepping through the whole HSB colour space in equal increments.

(d) What would the hierarchy look like after the Awake method has been called?



(2 marks)

(e) How would you procedurally animate the caterpillar so that torque is applied to to each segment in sequence?

(5 marks)

Solution
Make a float variable to hold the current segment and a speed variable. Should be a float so you can add speed * Time.delta time in Update.
current should wrap around so that it reaches the last segment it resets to 0
Apply torque around the current rigidbody's right vector

Question 4

(a) In relation to digital audio, explain the following terms: sample rate, resolution, frame size, spectrum, bin width.

Solution:
How many samples are taken per second. Measured in Hz.

Resolution:
The size in of each sample. Typically 8, 16, 24 or 32 bits.

Frame size:
The number of samples in each frame of audio to be analysed in an FFT

Spectrum:
The output of the FFT algorithm. A vector of the energies in each of the frequency bands.
Bin width:
The frequency range of each element of the spectrum

(5 x 2 marks)

(b) Figure 3 shows an extract from a Unity C# script that visualises the frequency spectrum of an AudioSource.

```
void CreateVisualisers()
{
    float theta = (Mathf.PI * 2.0f) / (float)AudioAnalyzer.frameSize;
    for (int i = 0; i < AudioAnalyzer.frameSize; i++)
    {
```

```
            Vector3 p = new Vector3(
                Mathf.Sin(theta * i) * radius
                , 0
                , Mathf.Cos(theta * i) * radius
                );
            p = transform.TransformPoint(p);
            Quaternion q = Quaternion.AngleAxis(theta * i * Mathf.Rad2Deg, Vector3.up);
            q = transform.rotation * q;

            GameObject cube = GameObject.CreatePrimitive(PrimitiveType.Cube);
            cube.transform.SetPositionAndRotation(p, q);
            cube.transform.parent = this.transform;
            cube.GetComponent<Renderer>().material.color = Color.HSVToRGB(
                i / (float)AudioAnalyzer.frameSize
                , 1
                , 1
                );
            elements.Add(cube);
        }
    }

    // Update is called once per frame
    void Update () {
        for (int i = 0; i < elements.Count; i++) {
            elements[i].transform.localScale = new Vector3(1, 1 + AudioAnalyzer.spectrum[i] *
scale, 1);
        }
     }
```

Figure 3

(c) In relation to the code answer the following questions:

(i) What shape will the generative visual have? How is the position of each segment in the visual calculated?

Solution:

It will be a bunch of cubes arranged in a circle. The script uses the unit circle (sin & cos), multiplied by the radius. The number of cubes is determined by framesize of the audiosource. The positions are calculated in local space and transformed relative to the container GameObject using transformPoint.

(ii) How is the orientation of each segment calculated?

Solution:

As a quaternion, where the angle is the index * the angle gap between adjacent segments. And the axis is the world up vector. Each segment will orient to the centre. This is then rotated by the container game object, (multiplied by it's quaternion)

(iii) How is the colour of each segment determined? What will the colour look like?

Solution:

By spanning the hue in the hue, saturation and value space (HSV). The hue will go from 0 – 1 in increments determined by the number of segments. The saturation and value indicate the amount of gray and the brightness of the colour. These are set to 1 (full). The colour will span all the colours of the rainbow from red back to red around the colour wheel.

(i) What aspect of the visual will be affected by audio? How would you improve the visual so that it was more responsive to the audio characteristics of music?

Solution:

The height of each of the segment will be affected by the amount of energy in each of the frequency bands of the spectrum. You could use logarithmic bands, where each band is twice the size of the previous. This more is more representative of perceptual frequency bands such as base, sub base, mid range etc.