CT3536 Marking Scheme, 2019-20

Q.1.

(i)

The Game Loop operates repeatedly, performing the core game tasks such as moving objects and re-drawing the screen [2]
The MonoBehaviour class provides a number of methods which are automatically called by the Unity engine as the Game Loop operates: e.g. Update(), OnCollisionEnter(), FixedUpdate(). [3]

(ii)

Coroutines are Monobehaviour methods which can be paused for various amounts of time [3]
Unity internally maintains a list of paused Coroutines and the Game Loop processes this each frame and resumes those whose pause time has elapsed. [2]

(iii)

Update method. [1]
transform.forwards as one argument to RotateTowards(). [1]
(Player.activePlayer.transform.position – transform.position).normalized as the other argument. [1]
Scaling rotation amount with Time.deltaTime. [1]
Assignment of result to zombie object via transform.LookAt or similar. [1]

(iv)

transform.forwards as one argument to Vector3.Dot(). [1]
(Player.activePlayer.transform.position – transform.position).normalized as the other argument. [1]
Testing for 'almost straight' via a dot product result of say >=0.8. [1]
Updating transform.position by some value * transform.forward [1]
Scaling movement amount with Time.deltaTime. [1]

Q.2.

Write technical notes on each of the following:

(i)

Add a Text or TextMeshPro text object to a Canvas. [2]
Obtain a reference to this object to make it accessible to the code updating the score [2]
Update the object's .text property whenever the score changes [1]

(ii)

Data on the Heap and Stack. [1]
Objects, strings, structs and simple data types. [1]
Mark and Sweep garbage collection. [1]
StringBuilder and generally minimizing objects being instantiated and destroyed during times when high frame rate is required; e.g. Object Pool pattern. [2]

(iii)

Definition of State Machine. [2]
Clear separation of logic at different times [2]
Example(s). [1]

(iv)

Screen space: on-screen pixels (2D). [1.5]
Viewport space: normalized on-screen position (2D). [1.5]
World space: position in the virtual world (3D). [1]
Camera class transforms between these spaces, according to the viewpoint of the camera. [1]

Q.3.

(i)

Collider is a component that defines a collision shape for use with physics. [1]
Typical use: to define physical bodies which will collide and respond, e.g. a ball and a floor. [1]
Trigger is a collider whose 'isTrigger' property is true (no physical collision response) [1]
Typical use: when you want your code to know when an object has moved into a region of space, e.g. a trap trigger in a dungeon. [1]
Interaction callbacks such as OnCollisionEnter() and OnTriggerEnter(). [2]

(ii)

OnCollisionEnter() happens when an object's collider touches another object's collider
Collision.contacts is an array of contact points. [2]
Iterate this array. [1]
Instantiate a 'spark' object at each contact's .point (which is a Vectror3 world position) [1]
Orient each 'spark' object to face towards the direction indicated by each contact's .normal (which is a Vector world direction). [2]

(iii)

transform.Rotate() [1]
Correct use of Space.World [1]

Calculation of difference vector. [0.5]
Calculation of direction vector. [0.5]
Multiplying direction by 7f and adding this to transform.position. [2]

transform.position = new Vector3(). [1]
Correct use of transform.forward or transform.TransformDirection(). [2]

Q.4.

(i)

Perform a transform.RotateAround() by a small amount each frame. [2]
Correct arguments (point, axis, angle). [2]
Update() method. [1]
No added components needed [2]

(ii)

Reference to planet GameObject or its Transform. [2]
Correct transform.RotateAround() arguments [3]

(iii)

Inspector variable for rotation parent GameObject or its Transform. [1]
Inspector variable for axis of rotation (Vector3). [2]
Inspector variable for speed of rotation (float). [1]
Update() method [1]
Correct transform.RotateAround() arguments, including use of Time.deltaTime with speed of rotation [3]

Q.5.

(i)

Raycast: source point and direction. [2]
Testing for intersecting objects in the physics world [3]
Raycast downwards from a point inside the character, to a point just below their feet [2]
True result means they're standing on something [1]
Correctly written Physics.Raycast(). [2]

(ii)

FixedUpdate() method. [1]
Raycast to test for grounded [2]
Input.GetKey() to test left/right/up keys [3]
Rigidbody2D.AddForce() or directly setting Rigidbody.velocity for left/right (must not remove existing up/down movement). [2]
Rigidbody2D.AddForce() or directly setting Rigidbody.velocity for jump, but only if grounded (must not remove existing left/right movement). [2]