INSTITUTE OF TECHNOLOGY BLANCHARDSTOWN

Academic term	2013-14
Year of study	2
Semester	SEMESTER TWO
Date of examination	
Time of examination	

Programme code	Programme title	Module code
BN 002	Higher Certificate in Computing in Information Technology	COMP H2033
BN 013	Bachelor of Science in Computing in Information Technology	COMP H2033
BN 104	Bachelor of Science (Hons) in Computing in Information Technology	COMP H2033

Module title	Interactive Multimedia SAMPLE EXAM - 2014 (given to students April 2014)

Internal Examiner(s)	**Dr. Matt Smith**
External Examiner(s)	**Dr. Tom Lunney, Mr. Michael Barrett**

Instructions to candidates:

1.	To ensure that you take the correct examination, please check that the module and programme which you are following is listed in the table above.
2.	This is a 3-hour, closed-book, practical examination. Your task is to create a 3D game from scratch.
3.	The total number of marks achievable in this examination is 100 marks. Answer ALL questions..
4.	Question 1 involves creating each scene. Questions 2 – 6 involve adding features to the game playing scene. The questions are presented in the recommended order of implementation, but you may wish to implement features in a different order.
5.	The Lecturer has provided you with useful common scripts, and all image and sounds files required beyond the standard Unity packages. Full colour screenshot images are also provided.
6.	Appendices provide Unity C# reference information and guidelines for code quality. Greyscale screenshots of each scene can also be found in the Appendices.
7.	**TECHNICAL NOTES** 1. Log onto the PC with your ITB network account. 2. The files the Lecturer has provided for the exam are at the following location, and you should save all your work in this location: **`K:\Exams`** 3. It is important to save your work regularly in case of power loss to your PC. 4. If you have a Technical issue accessing your network exam folder please ask the Exam Invigilator for assistance, who in turn can request Computer Services assistance if required. 5. At the end of the exam, ensure that all your answer files have been saved, and that you have quite all applications before logging off 6. You may not take a copy of your exam answers away with you. After results have been published you may request to see a copy of your work.

DO NOT TURN OVER THIS PAGE UNTIL TOLD TO DO SO

Name of game:

- Yellow Brick Road

Core mechanics (of fully completed game):

- To win the game:
  - Collect the 2 gold bricks
  - Then cross the bridge to get to the teleport cylinder
- You lose the game when:
  - all lives are lost
- Pickups
  - Gold bricks

Special features:

- when the 2 gold bricks have been collected, an animating arrow appears, and a bridge is created to the teleport, and music starts to loop
- player loses a life when step off the higher ground, or touch the rotating eyeball sphere
  - if they touch the rotating eyeball sphere a death sound is played

Scene list:

- scene 0 (main menu)
  - with an appropriate rollover button that make the game play scene number 1
  - with an appropriate rollover button that make the game play scene number 4
- scene 1 (level1 playing)
- scene 2 (game lost)
  - with an appropriate rollover button that make the game play scene number 0
- scene 3 (game won)
  - with an appropriate rollover button that make the game play scene number 0
- scene 4 (instructions)
  - with an appropriate rollover button that make the game play scene number 0

Question 1 (create each scene) **[15 marks]**

Create the following scenes (refer to screenshots and scene list in high level game design document):

- scene 0 (main menu)
- scene 3 (level1 playing)
    - create a scene containing a terrain for which:
        - good dirt texture applied to the whole terrain
        - a high raised sort of "Q" shape area (textured with yellow bricks)
        - a high raised "full stop" shape in one corner (textured with yellow bricks)
    - add the dawn-dusk skybox
    - add a 3rd person controller as follows
        - have its initial position on one of the 4 high sides
        - have a yellow spotlight shining down around its starting position
    - add 2 'walls' (textured squashed cubes) in the corner where the player starts
- scene 2 (game lost)
    - also this scene should play the lost game music audio clip continuously
- scene 3 (game won)
- scene 4 (instructions)

Question 2 **[20 marks]**

2a) Create two gold brick cubes as follows:

- cubes, textured with the yellow bricks
- tagged 'Gold', and destroys itself when hit
- positioned at the other end of the raised "Q" area from where the player begins

(4 marks)

2b) Create a bridge prefab as follows:

- create a stretched, textured cube
- position it to be like a bridge between where the user starts and the 'full stop' raised area in the other corner
- turn this bridge into a prefab (and delete it from the scene)

(5 marks)

2c) Create an animated eyeball game object as follows:

- create a material tinted red
- create an eyeball object (large white sphere, containing a smaller sphere with red material)
    - position it in between the start and 'hole' of the "Q" shape
- animate the eyeball object to rotate
- tag the object 'Eyeball', and destroys itself when hit

(5 marks)

2d) Create an animated arrow as follows:

- create arrow cube textured with the arrow image
- position it below ground, near where the player starts (with arrow pointing towards full stop area
- add loop animation (move above ground and towards finish spot)
    - the animation should loop
    - the animation should NOT play automatically when the scene loads

(5 marks)

Question 3 **[25 marks]**

3a) Implement the following features:

- when the player collides with the 'eyeball object, make that object disappear
    - and causes the dearth sound music to play once

(5 marks)

3b) Implement the following feature:

- each time the player hits a gold cube pickup, add 1 to a total, and display the total numbers of bricks to the user as text using GUILayout, e.g.:
    - "bricks: 1"
- when the player has collected 2 bricks make:
    - the bridge appear
    - start playing the arrow animation
    - start playing the music (once only)

(8 marks)

3c) Implement the following features:

- rather than the text showing how many gold bricks the player has collected, make your GUI display the appropriate image to show the number collected (0, 1, 2)

(5 marks)

Question 4 **[15 marks]**

4a) Implement the following features:

- the player starts with 2 lives
- the number of lives is displayed with the corresponding 'lives_left' image
- if the player lives gets below 0 then the game goes to the game lost scene

(5 marks)

4b) Implement the following features:

- player loses a life if they hit the 'eyeball' object
- player loses a life if they move off the high ground (if they height is too low)

(5 marks)

4c) Implement the following features:

- a countdown timer is running and displayed, counting down from 20
- player loses a life if the timer gets below 1
- when player loses a life, the timer restarts at 20 and they are respawned back to their start position

(5 marks)

Question 5 etc. **[??? marks]**

- when player hits teleport cylinder, go to game won screen
    - modify it so that game only won if music still playing when player gets to tetelport
- player can only wind
- etc. etc.

APPENDIX: GameObject reference

GameObject

Transform transform

Component attached to this GameObject (null if there is none attached).

GUIText guiText

Component attached to this GameObject (null if there is none attached).

GUITexture guiTexture

Component attached to this GameObject. (null if there is none attached)

Collider collider

Component attached to this GameObject (null if there is none attached).

string tag

The tag of this game object.

string name

The name of the object.

AudioSource audio

Component attached to this GameObject (null if there is none attached).

Renderer renderer

Component attached to this GameObject (null if there is none attached).

Rigidbody rigidbody

Component attached to this GameObject (null if there is none attached).

Camera camera

Component attached to this GameObject (null if there is none attached).

Light light

Component attached to this GameObject (null if there is none attached).

Animation animation

Component attached to this GameObject (null if there is none attached).

ConstantForce constantForce

Component attached to this GameObject (null if there is none attached).

NetworkView networkView

Component attached to this GameObject. (null if there is none attached)

HingeJoint hingeJoint

Component attached to this GameObject (null if there is none attached).

ParticleEmitter particleEmitter

Component attached to this GameObject (null if there is none attached).

APPENDIX: GameObject C# Code examples

AudioSource object inside an object with a Trigger

```
// play sound when something comes into our collider
private void OnTriggerEnter()
{
      audio.Play();
}
```

AudioSource object inside an object with a Trigger

```
// public image variable, so can be changed in Inspector
public Texture2D rolloverImage;

// show different image when mouse over object
private void OnMouseOver()
{
      guiTexture.texture = rolloverImage;
}
```

Change score according to object 'tag'

```
// show different image when mouse over object
private void OnTriggerEnter(Collider hitObjectCollider)
{
      string tag = hitObjectCollider.tag;

      if( "goodApple" == tag )
      {
            score += 10;
      }
      else if( "blueKey" == tag )
      {
            isHoldingBlueKey = true;
            Destroy( hitObjectCollider.gameObject );
      }
}


// alteranative way to work with tags – CompareTag() method
private void OnTriggerEnter(Collider hit)
{
      if(hit.CompareTag("goodApple")
      {
            score += 10;
      }
}
```

APPENDIX: Collider reference

GameObject

Collider collider

Useful properties

- bool isTrigger

Useful events

- OnTriggerEnter() / OnTriggerEnter(Collider hitObjectCollider)
    - Can ignore parameter if not required for game logic
    - Event generated when another object's rigidbody or collider ENTERS the collider of this object
- OnTriggerExit() / OnTriggerExit(Collider hitObjectCollider)
    - Can ignore parameter if not required for game logic
    - Event generated when another object's rigidbody or collider EXITS the collider of this object
- OnTriggerStay() / OnTriggerStay(Collider hitObjectCollider)
    - Can ignore parameter if not required for game logic
    - Event generated ALMOST ever frame while another object's rigidbody or collider IS INSIDE the collider of this object
- Note
    - Also 'Collision' messages, rather than Trigger we need to work with impact data (direction / forces etc.)

Summary

- Most used for detecting collisions between player controlled characters (first or third) and objects in the game world
    - But can be used for objects colliding due to physics applied to rigidbodies …

Addition information/suggestions:

- Ensure the Is Trigger checkbox is ticked for each item you want the player's character to collide with
- Objects whose Is Trigger checkbox is ticked generate an OnTriggerEnter() message when another object enters their 'collider'
    - The OnTriggerEnter() message is sent BOTH to the object that is hit, and the object that moved into it
    - For example, if the player's man bumps into a sphere, both the sphere object, and the man's character object are both send an OnTriggerEnter() message

APPENDIX: Collider C# Code examples

AudioSource object inside an object with a Trigger

```
// play sound when something comes into our collider
private void OnTriggerEnter()
{
      audio.Play();
}
```

ensure do not send Play message if sound already playing

```
private void OnTriggerEnter()
{
      if( !myAudioSource.isPlaying )
      {
         myAudioSource.Play();
      }
}
```

Change score according to object 'tag'

```
// show different image when mouse over object
private void OnTriggerEnter(Collider hitObjectCollider)
{
      string tag = hitObjectCollider.tag;

      if( "goodApple" == tag )
      {
            score += 10;
      }
      else if( "blueKey" == tag )
      {
            isHoldingBlueKey = true;
            Destroy( hitObjectCollider.gameObject );
      }
}
```

Open (Destroy) door if holding blue key

```
// show different image when mouse over object
private void OnTriggerEnter(Collider hitObjectCollider)
{
      string tag = hitObjectCollider.tag;

      if( ( "blueDoor" == tag ) && isHoldingBlueKey )
      {
            // open the door
            Destroy( hitObjectCollider.gameObject );
      }
}
```

APPENDIX: GUITexture reference

GameObject

GUITexture guiTexture

Useful properties

- Texture2D texture
- Color color

Useful methods

- OnMouseOver()
  - When mouse over object
- OnMouseExit()
  - When mouse moves away from object
- OnMouseUp()
  - When mouse clicked on object

Summary

- GameObjects containing GUITextures can display an image to the user
- When used with mouse events, can turn into a rollover 'button' …

Warning(s):

- If change texture (image) to one of a different size, then new image will be SQUASHED or STRETCHED to fit in space of old image …

Addition information/suggestions:

- Have a script with an ARRAY of textures, and can loop through them to give impression of video or animation …
- Mouse events like OnMouseUp() can also be used with 3D game objects

APPENDIX: GUITexture C# Code examples

Change color of GUITexture upon being created

```
// use pre-defined green color
private void Awake()
{
      guiTexture.color = Color.green;
}
```

Change texture (image) when mouse rolled over

```
// variable for rollover image (set in Inspector)
public Texture2D rolloverImage;

// change image when mouse over
private void OnMouseOver()
{
      guiTexture.texture = rolloverImage;
}
```

Make Unity load level integer when clicked

```
// default to zero (can change in Inspector)
public int levelNumToLoad = 0;

// load scene/level when clicked by mouse
private void OnMouseUp()
{
      Application.LoadLevel(levelNumToLoad );
}
```

APPENDIX: AudioSource reference

GameObject

AudioSource audio

Useful properties

- AudioClip clip
- float volume
- bool isPlaying (read only)
- bool loop

Useful methods

- Play()
- Stop()
- PlayOneShot()

Summary

- Objects that contain an AudioSource component can play sound files (AudioClips). The sound is picked up by an AudioListener (usually the one in the Main Camera).

Warning(s):

- If the object containing the AudioSource component is far away from the AudioListener, it will not be heard, even if the sound is playing

Addition information/suggestions:

- avoid sending a Play() message to an already playing clip – make use of the read only property **isPlaying**, e.g.
- prefab + play on awake
    - have an AudioSource component inside a Prefab (e.g. an explosion or enemy character)
    - tick Play On Awake
    - when prefab instance is created sound will play – no scripting needed
- animation event
    - have a function that changes the sound file (AudioClip) and then makes it play

APPENDIX: AudioSource C# Code examples

AudioSource object inside an object with a Trigger

```
// play sound when something comes into our collider
private void OnTriggerEnter()
{
      audio.Play();
}
```

Display a button and send message to another object that contains an audio source – so need public reference

```
/** public reference to AudioSource component in another object */
public AudioSource myAudioSource;

/** display button to send  Play() message */
private void OnGUI()
{
      bool playButtonWasClicked = GUILayout.Button("Play");

      if( playButtonWasClicked )
      {
            myAudioSource.audio.Play();
      }
}
```

ensure do not send Play message if sound already playing

```
private void OnTriggerEnter()
{
      if( !myAudioSource.isPlaying )
      {
         myAudioSource.Play();
      }
}
```

FPS Camerascript – contains an audio source component ..

```
// drag sound clip to this public variable in the Inspector
public AudioClip crashSound;

// play crash sound if hit ANYTHING !
private void OnTriggerEnter()
{
        audio.PlayOneShot( crashSound );
}
```

Useful function for animation event - assumes an audio source component ..

```
private void PlayAudioClip(AudioClip clip){
        audio.PlayOneShot( clip );
}
```

APPENDIX: Code Quality Guidelines

All your code should demonstate correctness and high quality, consistent naming and layout

- layout and comments
    - Indentation
    - use of comments and blank links to layout code
- class, method and property identifiers, and object tags
    - meaningful names
    - use of upper and lower camel-case, and to indicate variables, methods, classes and tags
- Appropriate use of keywords ‘public’ and ‘private’ for all methods and properties
- Efficient code
    - for example by caching references when objects are created
    - only having actions in methods Update() and OnGUI() that require execution every frame
- Well behaved OO techniques, such as
    - private variables and public getters and setters
    - avoiding inappropriate use of static properties and methods
- Avoid ‘hard coding’ numbers into code
    - use well named constants to give meaning to values used in methods

SCREENSHOT – Scene 0 main menu

Etc.