NUI Galway
OÉ Gaillimh

# ***Semester 1 Examinations 2013/ 2014***

| | |
|---|---|
| **Exam Code(s)** | 4BCT, 3BA, 1SD, 1MDM |
| **Exam(s)** | 4th Year B.Sc. (CS&IT)<br>3rd Year B.A. (Information Technology)<br>Higher Diploma in Applied Science (Software Design & Development)<br>Masters in Digital Media |
| **Module Code(s)** | CT404, CT336 |
| **Module(s)** | Graphics and Image Processing |
| Paper No. | |
| Repeat Paper | |
| External Examiner(s) | Prof. L. Maguire |
| Internal Examiner(s) | Prof. G. Lyons<br>Dr. M. Madden<br>* Dr. S. Redfern |
| ***Instructions:*** | Answer any three questions.<br>All questions carry equal marks. |
| ***Duration*** | 2 hours |
| **No. of Pages** | 7 |
| **Discipline(s)** | Information Technology |
| **Course Co-ordinator(s)** | |

**Requirements**:

| | | | | | |
|---|---|---|---|---|---|
| MCQ | Release to Library: | Yes | X | No | |
| Handout | | | | | |
| Statistical/ Log Tables | | | | | |
| Cambridge Tables | | | | | |
| Graph Paper | | | | | |
| Log Graph Paper | | | | | |
| Other Materials | | | | | |
| Graphic material in colour | Yes | X | No | | |

**PTO**

**Q.1. (Graphics)**

(i) Consider the following algorithm, which may be used to determine, in a 2-dimensional coordinate system, which side of a straight line a point lies on:

- Given a line segment (x0,y0) to (x1,y1), and a point (x,y)
- Compute: s = (y - y0)*(x1 - x0) - (x - x0)*(y1 - y0)
  - If s is less than 0 then (x,y) is to the right of the line segment
  - If s is greater than 0 then (x,y) is to the left of the line segment
  - If s is equal to 0 then (x,y) lies on the line segment

Explain, with the use of diagram(s), how this function can be used for collision detection in a 2D vector game such as 'Asteroids'. You can assume that game objects are defined as a number of straight lines between vertices. [6]

Discuss the effect that concavities in a game object would have on this approach [2]

(ii)
Explain in general the keyframe approach to animation in computer graphics, and explain specifically its use in X3D, referring to the TimeSensor, Transform, OrientationInterpolator and PositionInterpolator nodes in your answer. [5]

(iii)
Consider the HTML5/Javascript code below. This code handles keypresses on the document by invoking the `draw()` function. The key that was pressed is sent as a single-character string. Write suitable Javascript code to manipulate the Canvas context object in order to:

- Clear any existing graphics [1]
- Draw a rectangle whose position changes by 1 pixel each time one of these four keys are pressed: (W=up, A=left, D=right, S=down) [4]
- Disallow the rectangle from moving outside the bounds of the Canvas [2]

```
<html>
 <head>
  <script>
        function attachEvents() {
                document.onkeypress = function(event) {
                        var key = String.fromCharCode(event.keyCode || event.charCode);
                        draw(key);
                }
        }
        function draw(key) {
          var canvas = document.getElementById("canvas");
          var context = canvas.getContext('2d');

          // to do: clear existing graphics on the Canvas

          // to do: draw a rectangle whose position changes based on key
          // that was pressed (W=up, A=left, D=right, S=down)

          // to do: don't allow the rectangle to pass outside of the
          // bounds of the Canvas

        }
  </script>
 </head>
 <body onload="attachEvents();">
   <canvas id="canvas" width="300" height="300"></canvas>
 </body>
</html>
```

**Q.2. (Graphics)**

(i) *The Z-Buffer Algorithm*

- With regard to the rendering of 3D graphics, discuss in detail the operation of the Z-Buffer Algorithm. [3]
- Explain what it means for this algorithm to be an 'image space' technique. [2]
- How might the Z-Buffer be useful for post-rendering effects such as depth blur? [2]

(ii)

With regard to 3D computer graphics, explain with use of diagram(s), the fundamental differences between parallel projections and perspective projections. Your explanation should make it clear how 3D objects are projected onto a 2D plane to form an image on the screen. [6]

(iii)

Consider the HTML5/Javascript code below. This code presents the user with two text boxes into which they type numbers. A button labelled 'Redraw' is presented and when this is clicked, the `draw()` function is invoked.

Write suitable Javascript code to manipulate the Canvas context object in order to:

- Clear any existing graphics [1]
- Draw a rectangle of width 50 pixels and height 50 pixels, at the (x,y) position as specified by the numbers entered by the user into the two text boxes [3]

Modify the program so that, when a rectangle is drawn, all previous rectangles are also re-drawn (each at width 20 pixels, height 20 pixels) [3]

```
<html>
<head>
<script>
function draw() {
	var canvas = document.getElementById("canvas");
	var ctx = canvas.getContext("2d");

	// obtain the x and y values entered by the user
	var x = document.getElementById("xPosition").value;
	var y = document.getElementById("yPosition").value;
}
</script>
</head>

<body>
<form>
	<input id='xPosition' value='30'><BR>
	<input id='yPosition' value='40'><BR>
	<input type=button value='Redraw' onclick='draw();'>
</form>

<canvas id="canvas" width="600" height="600"></canvas>
</body>
</html>
```

**Q.3. (Graphics)**

(i)
Consider the 3D model of a road barrier depicted below, which is constructed from 7 separate pieces (4 legs, 2 small struts, and 1 main barrier piece). Write X3D code to produce this model as accurately as possible. Note that:

- The legs and struts are not rectangular in shape and should therefore be created using Extrusion. Please sketch your cross-section calculations.
- You can assume that you have been provided with two suitable texture files "blue.jpg" for the legs and struts, and "yellow.jpg" for the main barrier
- Some useful X3D nodes are summarised on the final page of this exam paper

[10]


(ii)
Discuss, with the use of diagram(s) in each case, the following 3D graphics techniques:

- Flat (Lambert) Shading [2]
- Smooth (Gourard) Shading [2]
- Dynamic lighting using Point Lights, Directional Lights, and Ambient Lights [4]
- Pre-calculated lighting using Radiosity [2]

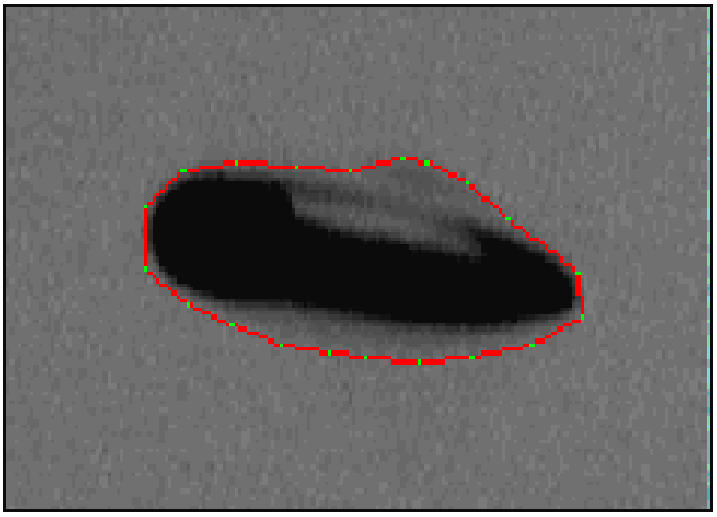
**Q.4. (Image Processing)**

(i)
Convolution (also called 'filtering') is one of the most fundamental low-level image processing techniques.

- Describe, with use of a diagram, the convolution algorithm. [3]
- Present a convolution kernel suitable for noise reduction. Explain how this kernel produces the desired result, assuming a greyscale image with pixel values in the range 0 to 255. [3]
- Present a convolution kernel suitable for edge detection. Explain how this kernel produces the desired result, assuming a greyscale image with pixel values in the range 0 to 255. [4]

(ii)
Considering the image processing technique called 'active contours' or 'snakes':

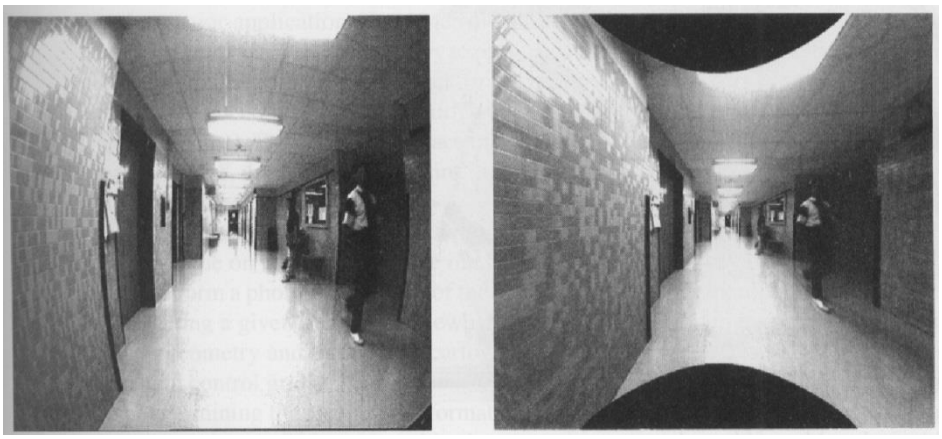
- Outline the basic algorithm used by the technique, using the following terms in your answer: (a) snaxel, (b) energy function, (c) global optimisation [4]
- Why is the active contours approach particularly suitable for tracking objects in image sequences? [2]
- Present a suitable set of optimisation constraints (energy factors) for accurately tracing the outline of a bubble such as the one shown below, using active contours. [4]



**Q.5. (Image Processing)**

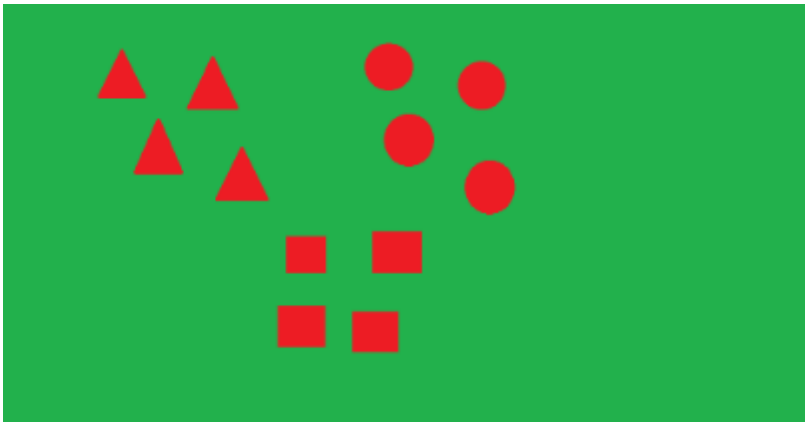
(i)
'Camera decalibration' is a technique for geometric correction of images which is often employed when sources of geometric error are poorly understood. With regard to camera decalibration:

- Outline the use of reference images such as grids of dots to construct and apply geometric corrections to images captured with a wide-angle lens. Use the terms 'control points', 'pixel filling', and 'bilinear interpolation' in your answer. [7]
- Explain why would you expect a reference image to be constructed with black markings on a white background (or white markings on a black background) [3]



(ii)
Devise, discuss and defend a multi-step image processing algorithm for accurately counting the number of rectangles, circles, and triangles in an image such as the one shown below.

[10]



## Some useful X3D nodes:

| Node | Important Fields and Nested Nodes |
|---|---|
| Shape | Nested Nodes: Appearance, Geometry Nodes (Box, Sphere, Cone, Cylinder, Text, Extrusion, etc.) |
| Appearance | Nested Nodes: Material, ImageTexture |
| Material | Fields: diffuseColor, specularColor, emissiveColor, ambientIntensity, transparency, shininess |
| ImageTexture | Fields: url |
| Transform | Fields: translation, rotation, scale, center.<br>Nested Nodes: Other Transforms, Shapes, Sensors |
| TimeSensor | Fields: enabled, startTime, stopTime, cycleInterval, loop |
| PositionInterpolator | Fields: key, keyValue |
| OrientationInterpolator | Fields: key, keyValue |
| Extrusion | Fields: crossSection, spine, scale, orientation, beginCap, endCap, creaseAngle |
| Box | Fields: size |
| Sphere | Fields: radius |
| Cylinder | Fields: radius, height, side, top, bottom |
| Cone | Fields: height, bottomRadius, side, bottom |
| PointLight | Fields: on, location, radius, intensity, ambientIntensity, color, attenuation |
| ROUTE | Fields: fromNode, fromField, toNode, toField |

## Some useful methods/properties of the Canvas 2D Context object:

| Method/Property | Arguments/Values | Notes |
|---|---|---|
| fillRect | (Left, Top, Width, Height) | Draw a filled rectangle |
| beginPath | None | Start a stroked path |
| moveTo | (X, Y) | Move the graphics cursor |
| lineTo | (X, Y) | Draw a line from graphics cursor |
| stroke | None | End a stroked path |
| fillStyle | ="rgb(R,G,B)" | Set fill colour |
| strokeStyle | ="rgb(R,G,B)" | Set line colour |
| save | None | Save the current coordinate system |
| restore | None | Restore the last saved coord system |
| translate | (X,Y) | Translate the coordinate system |
| rotate | (angle) | Rotate the coordinate system clockwise, with angle in radians |
| scale | (X,Y) | Scale the coordinate system independently on the X and Y axes |