Computer Graphics II: Tools and Techniques

P. Healy

CS1-08
Computer Science Bldg.
tel: 202727
patrick.healy@ul.ie

Autumn 2020–2021
Outline

1. Administrative Details
   - Meeting Times
   - General Issues
   - Assessment

2. To Do

3. Syllabus

4. Review Material
   - Vectors
Outline

1. **Administrative Details**
   - Meeting Times
   - General Issues
   - Assessment

2. **To Do**

3. **Syllabus**

4. **Review Material**
   - Vectors
Lectures / Labs / Tutes

Lecture Hours:  
Mon. 10h00  
Tue. 13h00  

Lab  
Mon. 16h00  
→ 4A  

Tute  
Mon. 13h00  

Wks\{3,7,11\} On Campus  
CS-304B, Wks\{3,7,11\}  
extcept Wks\{3,7,11\}  

5 contact hours → 5 non-contact hours
Outline

1. **Administrive Details**
   - Meeting Times
   - General Issues
   - Assessment

2. **To Do**

3. **Syllabus**

4. **Review Material**
   - Vectors
Attendance

- Attendance at all lectures and labs / tutes is expected
- Handing up all assigned lab exercises is a good idea
- Handing up other people’s work is a serious

What’s this about? Why is this on the slide?
All lectures, homeworks, past exams, etc. also can be found on the class home page:

garryowen.csisdmz.ul.ie/~cs4085/ (also goo.gl/WUYL)

Class lists and attendance records will also be available here
All lectures, homeworks, past exams, etc. also can be found on the class home page: 
garryowen.csisdmz.ul.ie/~cs4085/ (also goo.gl/WUYL)

Class lists and attendance records will also be available here
Outline

1. Administrative Details
   - Meeting Times
   - General Issues
   - Assessment

2. To Do

3. Syllabus

4. Review Material
   - Vectors
Assessment Instruments

- Programming Project (i): 15% Week04
- Programming Project (ii): 40% Week07
- Programming Project (iii): 25% Week11
- In-class participation, quizzes: 20%

- Tutes, labs start next week
## Grade Bands

<table>
<thead>
<tr>
<th>Grade</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>0 – 29</td>
</tr>
<tr>
<td>D2</td>
<td>30 – 34</td>
</tr>
<tr>
<td>D1</td>
<td>35 – 39</td>
</tr>
<tr>
<td>C3</td>
<td>40 – 47</td>
</tr>
<tr>
<td>C2</td>
<td>48 – 51</td>
</tr>
<tr>
<td>C1</td>
<td>52 – 55</td>
</tr>
<tr>
<td>B3</td>
<td>56 – 59</td>
</tr>
<tr>
<td>B2</td>
<td>60 – 63</td>
</tr>
<tr>
<td>B1</td>
<td>64 – 71</td>
</tr>
<tr>
<td>A2</td>
<td>72 – 79</td>
</tr>
<tr>
<td>A1</td>
<td>80 – 100</td>
</tr>
</tbody>
</table>
Reading List


3. *David H. Eberly 3D Game Engine Design (2nd ed.)*


Reading List

**Reading List**

3. *David H. Eberly* 3D Game Engine Design (2nd ed.)
To Do (for You)

- Don’t forget to register online at http://www.si.ul.ie
- Sign up for Study Skills Workshop (Week02) via the First 7 Weeks page on Facebook, www.Facebook.com/first7weeks
Aims and Objectives

- Increase competence of student in the area of modern real-time computer graphics. This includes usage of Content Creation Suites, 3D Engines and combining available tools into a working tool chain. This is a follow-on module to CS4815 introducing more advanced graphics techniques and special effects.

- After finishing the course the student will gain competence in use of various tools to set up a tool chain for content creation and developing a 3D real-time application for content presentation.
Learning Outcomes

On successful completion of this module, students should be able to:

- Utilise various tools to set up a tool chain for content creation
- Manipulate files in varying graphical formats
- Combine basic modelling and animation techniques in modern graphics systems
- Apply their skills in developing a 3-D real-time applications for content presentation
- Implement basic tasks on a GPU
- Demonstrate an insight into the steps required to achieve realistic graphical content
Detailed Syllabus

- Vectors/Matrices Review
- Basic Modelling Techniques
- Quaternions
- Basic Animation Techniques
- Particle Systems
- Usage of Content Creation Suites
- Introduction to Real-Time 3D engines
- Scene Management Techniques
- Pixel/Vertex Shaders
Outline

1. Administrative Details
   - Meeting Times
   - General Issues
   - Assessment

2. To Do

3. Syllabus

4. Review Material
   - Vectors
Really nice series of videos on vectors, etc.
Vectors Review

- Vectors have **no** position; they are defined by their *length* and their *direction*; more precisely, they are defined as the **difference** between two points.

That said, at times it is often handy to place them at the origin; if we do this then a point, \( \mathbf{p} = (x, y)^t \), can be represented by the vector \( \mathbf{v} \) from the origin to \( \mathbf{p} \).
Vectors have **no** position; they are defined by their *length* and their *direction*; more precisely, they are defined as the *difference* between two points.

That said, at times it is often handy to place them at the origin; if we do this then a point, \( p = (x, y)^t \), can be represented by the vector \( \vec{v} \) from the origin to \( p \),

\[
\vec{v} = (x - 0, y - 0)^t = (x, y)^t
\]
Vectors Review (contd.)

**Vector Bases**

- Just as words are made up from the basic unit of language, letters, likewise vectors are composed of basic vectors.
- We call this set of basic vectors a *basis*.
- The vectors in a basis do not have to be orthogonal (at right angles to each other) but they usually are.
- The vectors in a basis do not have to be of unit length but they usually are.
- We will encounter several co-ordinate systems during the semester and each will actually be a vector basis; that is, any point in the co-ordinate system will be representable by some combination of the basis vectors.
Some maths textbooks use the notation that the vectors $\vec{x}, \vec{y}, \vec{z}$ are the three unit-length, orthogonal vectors in the basis; other books use the notation $\vec{i}, \vec{j}, \vec{k}$ as the basis vectors.

So if a vector, $\vec{v}$, is drawn pointing from $(2, 0)$ to $(0, 3)$

- in one notation it is $\vec{v} = -2\vec{x} + 3\vec{y}$;
- in another notation (book) it could be $\vec{v} = -2\vec{i} + 3\vec{j}$;
- and it could also be the vector $\vec{v} = (-2, 3)^t$.
**Vectors Review (contd.)**

**Arithmetic**

- Vector addition is “head to tail”:

- Vector subtraction is “head to head”:

- Scalar multiplication is a “fraction” of a vector and is a vector, itself; if $\mathbf{u} = (u_1, u_2, \ldots, u_n)^t$ then when $r$ is a real no. (a scalar), $r\mathbf{u} = (ru_1, ru_2, \ldots, ru_n)^t$: 
**Arithmetic**

- Vector addition is “head to tail”:
- Vector subtraction is “head to head”:

![Diagram showing vector addition and subtraction](image)

- Scalar multiplication is a “fraction” of a vector and is a vector, itself; if \( \vec{u} = (u_1, u_2, \ldots, u_n)^t \) then when \( r \) is a real no. (a scalar), \( r\vec{u} = (ru_1, ru_2, \ldots, ru_n)^t \):
Vectors Review (contd.)

**Arithmetic**

- Vector addition is “head to tail”:
- Vector subtraction is “head to head”:
- Scalar multiplication is a “fraction” of a vector and is a vector, itself; if \( \vec{u} = (u_1, u_2, \ldots, u_n)^t \) then when \( r \) is a real no. (a scalar), \( r\vec{u} = (ru_1, ru_2, \ldots, ru_n)^t \):

![Diagram of vector operations]
The **handedness** of a 3D co-ordinate system is crucial for keeping consistency between the three (mutually perpendicular) dimensions: given the positive directions of 2 of the axes, the handedness determines which way the positive third axis points.

In an RHCS where the normal ordering is \(x, y, z\) if the \(y\) axis points “up” and the \(x\) axis points right then the \(z\) axis would point out of the page.

OpenGL is a **right** handed co-ordinate system; Direct3D is a **left** handed co-ordinate system.
Handedness of co-ordinate systems

The **handedness** of a 3D co-ordinate system is crucial for keeping consistency between the three (mutually perpendicular) dimensions: given the positive directions of 2 of the axes, the handedness determines which way the positive third axis points.

Curl your **right** hand around the third axis grasping from positive $x$ to $y$, and your thumb will point in the positive $z$ direction; this assumes that normal ordering of axes is $x, y, z$.

In an RHCS where the normal ordering is $x, y, z$, if the $y$ axis points "up" and the $x$ axis points right then the $z$ axis would point out of the page.
Handedness of co-ordinate systems

The **handedness** of a 3D co-ordinate system is crucial for keeping consistency between the three (mutually perpendicular) dimensions: given the positive directions of 2 of the axes, the handedness determines which way the positive third axis points.

**Important:**
When we talk about **cross products** later remember that if the third axis is to point towards you then the ordering has got to be **anti-clockwise**.
Handedness of co-ordinate systems

The **handedness** of a 3D co-ordinate system is crucial for keeping consistency between the three (mutually perpendicular) dimensions: given the positive directions of 2 of the axes, the handedness determines which way the positive third axis points.

In an RHCS where the normal ordering is $x, y, z$ **if** the $y$ axis points “up” and the $x$ axis points right then the $z$ axis would point out of the page.

OpenGL is a **right** handed co-ordinate system; Direct3D is a **left** handed co-ordinate system.
The **handedness** of a 3D co-ordinate system is crucial for keeping consistency between the three (mutually perpendicular) dimensions: given the positive directions of 2 of the axes, the handedness determines which way the positive third axis points.

In an RHCS where the normal ordering is $x, y, z$ *if* the $y$ axis points “up” and the $x$ axis points right then the $z$ axis would point out of the page.

OpenGL is a **right** handed co-ordinate system; Direct3D is a **left** handed co-ordinate system.