

Assignment due date: Feb 10, 2017, 11:59pm

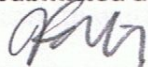
**Hand-in to be submitted electronically in PDF format with
code to the CDF server by the above due date**

Student Name (last, first): HIOE FELICIA

Student number: 1000526111

Student UTORid: hioefel

I hereby affirm that all the solutions I provide, both in writing and in code, for this assignment are my own. I have properly cited and noted any reference material I used to arrive at my solution and have not share my work with anyone else. I am also aware that should my code be copied from somewhere else, whether found online, from a previous or current student and submitted as my own, it will be reported to the department.



Signature

(Note: -3 marks penalty for not completing properly the above section)

Part 1 total marks: 50

Part 2 total marks: 50

Total: 100

This assignment has 2 main goals: First, to let you practice with geometry and transformations so that you gain familiarity with vector operations and achieve a clear understanding of how to translate Geometry into Linear Algebra. Secondly, it will let you practice writing a program that applies all the geometric concepts covered so far, and produces a 3D animated result using OpenGL.

Learning Objectives:

You will learn to manage points and vectors using vector operations and dot products.

You will practice with the use of transformations on points, and gain insight into how transformations affect points and sets of points.

You will understand the relationship between Linear Algebraic equations and geometric operations on points and vectors.

You will analyze simple equations of trajectories that may be found in animated objects. You will learn to write code that deals with geometric entities and transformations. You will learn to animate large sets of objects using simple behavioural rules.

Skills Developed:

Thinking in terms of geometry and vectors.

Translating Geometry into Linear Algebra and vice-versa.

Writing simple OpenGL code to display animated scenes.

Reference material:

The Lecture notes up to this point, found on the course website.

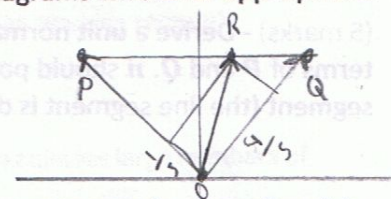
All the comments within Boids.cpp. Be sure to read carefully and understand what the code is doing and what you are supposed to implement **before** you start coding.

Your OpenGL reference text.

Part 1 - Written work: Be sure to provide clean, legible derivations and not omit any steps which your TA may need to fully understand your work. Use diagrams wherever appropriate.

A) Lines, points, vectors, and dot-products

1) Let a line segment be defined by the points P and Q .



(4 marks) - Let R be the point on the line segment that is four times as far from P as it is from Q . Let $\vec{p} = \overrightarrow{OP}$, $\vec{q} = \overrightarrow{OQ}$, $\vec{r} = \overrightarrow{OR}$. Show that $\vec{r} = \frac{1}{5}\vec{p} + \frac{4}{5}\vec{q}$ and draw a sketch validating this.

$$\begin{aligned}
 4|QR| &= |PR| \\
 4|\vec{r} - \vec{q}| &= |\vec{r} - \vec{p}| \\
 4|\vec{q} + k(\vec{p} - \vec{q}) - \vec{q}| &= |\vec{q} + k(\vec{p} - \vec{q}) - \vec{p}| \\
 4k|\vec{p} - \vec{q}| &= |(k-1)(\vec{p} - \vec{q})| \\
 4k|\vec{p} - \vec{q}| &= 1-k|\vec{p} - \vec{q}|
 \end{aligned}$$

$$\begin{aligned}
 4k &= 1-k \\
 k &= \frac{1}{5} \\
 \vec{r} &= \vec{q} + k(\vec{p} - \vec{q}) \\
 &= \vec{q} + \frac{1}{5}(\vec{p} - \vec{q}) \\
 &= \frac{4}{5}\vec{q} + \frac{1}{5}\vec{p}
 \end{aligned}$$

(3 marks) - Show the parametric form of the line segment formed by P and Q , with parameter t , starting from P in the direction of Q (that is to say, positive values of the parameter will give points along the line towards Q). Be sure to define everything explicitly.

Let the points in P be (x_0, y_0) ,
and the points in Q be (x_1, y_1) .
Then: $x = x_0 + t(x_1 - x_0)$ R be (x, y) .
 $y = y_0 + t(y_1 - y_0)$ or $R = P + t(Q - P)$
where $t \in [0, 1]$

(5 marks) Let the points have explicit representations $P(x_0, y_0)$, $Q(x_1, y_1)$ with respect to a coordinate system. Give a simple test to determine if an arbitrary point $R(x, y)$ is on the line segment or not.

- test cross product for alignment of
 $(y - y_0)(x_1 - x_0) - (x - x_0)(y_1 - y_0) = 0$
- test dot product for distance
 $(x - x_0)(x_1 - x_0) + (y - y_0)(y_1 - y_0) > 0$
- and
 $(x - x_0)^2 + (y - y_0)^2 < (x_1 - x_0)^2 + (y_1 - y_0)^2$

2)

(5 marks) - Derive a **unit normal vector** \vec{n} to the line segment from above, expressed in terms of P and Q . \vec{n} should point to the left with respect to the direction of the line segment (the line segment is directed from P to Q).

Let a line segment be defined by the points P and Q .
 (1 mark) - Let P be the point on the line segment that is four times as far from Q as it is from Q . Let $\vec{p} = 0.8\vec{P} + 0.2\vec{Q}$.
 (1 mark) - Let $\vec{q} = 0.2\vec{P} + 0.8\vec{Q}$.
 (1 mark) - Let $\vec{r} = 0.5\vec{P} + 0.5\vec{Q}$.
 (1 mark) - Let $\vec{s} = 0.5\vec{P} + 0.5\vec{Q}$.
 (1 mark) - Let $\vec{t} = 0.5\vec{P} + 0.5\vec{Q}$.

Diagram showing points P and Q with a line segment between them. A vector $\vec{n}_{P,Q}$ is shown pointing to the left of the segment.

Let $P(x_0, y_0)$
 Let $Q(x_1, y_1)$
 Normal vector: $(-(y_1 - y_0), x_1 - x_0)$
 Normalized normal vector:

$$\vec{n} = \frac{-(y_1 - y_0)}{\sqrt{(y_1 - y_0)^2 + (x_1 - x_0)^2}} \vec{i} + \frac{x_1 - x_0}{\sqrt{(y_1 - y_0)^2 + (x_1 - x_0)^2}} \vec{j}$$

(5 marks) - Using \vec{n} from 2), give an expression that computes the distance between an arbitrary point $R(x, y)$ and the line segment.

If $0 < \frac{\vec{PR} \cdot \vec{PQ}}{\|\vec{PR}\| \cdot \|\vec{PQ}\|} < 1$ and $0 < \frac{\vec{QR} \cdot \vec{QP}}{\|\vec{QR}\| \cdot \|\vec{QP}\|} < 1$

then distance = $\|\vec{n} \cdot \vec{PR}\|$

ELSE IF $0 < \frac{\vec{PR} \cdot \vec{PQ}}{\|\vec{PR}\| \cdot \|\vec{PQ}\|} < 1$

then distance = $\|\vec{QR}\|$

ELSE

distance = $\|\vec{PR}\|$

B: Transformations and transformation properties

(3 marks each) - Prove whether or not the following pairs of transformations commute.

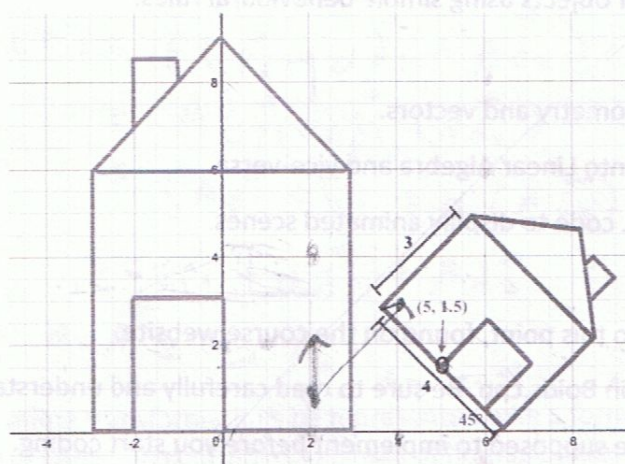
The transformations are general, 2D affine transforms (provide solutions on separate sheet)

- 1) Rotations and Translations No
- 2) Two rotations Yes
- 3) Translation, Reflection No
- 4) Shear wrt to x axis and uniform scaling No
- 5) Rotation, non-uniform scaling No

If not-commutative, a simple counter example will suffice. If commutative, algebraic proof is required.

C: Affine transformation properties

- 1) (5 marks) - Give the sequence of 2D affine transformations that maps the object in the left figure to the object in the right figure. You can use R, T, S, Sh, Re, to express rotation, translation, scaling, shear, and reflection operations respectively.



- Re on y-axis
- S to make house smaller
- T right by 4
- R on the lower right corner of the house by 45°

- 2) (5 marks) A triangle can be expressed in parametric form as:

$$T(\alpha, \beta) = \vec{p}_1 + \alpha \vec{d}_1 + \beta \vec{d}_2$$

provided that $\alpha + \beta \leq 1$ and $\alpha, \beta \geq 0$. Prove that under an invertible affine transform, the triangle remains a triangle.

Let $f(\vec{x}) = A\vec{x} + \vec{b}$ be an affine transformation.

$$\begin{aligned} f(\vec{x}) &= f(\vec{p}_1 + \alpha \vec{d}_1 + \beta \vec{d}_2) \\ &= A(\vec{p}_1 + \alpha \vec{d}_1 + \beta \vec{d}_2) + \vec{b} \\ &= (A\vec{p}_1 + \vec{b}) + \alpha(A\vec{d}_1) + \beta(A\vec{d}_2) \\ &= \vec{p}_1 + \alpha \vec{d}_1 + \beta \vec{d}_2 \end{aligned}$$

where $\alpha + \beta \leq 1$.

\therefore triangle remains a triangle

- 3) (3 marks) - Can affine transformations be represented with a formula where points can be represented in Cartesian coordinates instead of homogeneous coordinates? What is the main advantage of representing affine transformations in the same matrix form as general homographies?

Yes, affine transformations can be represented with a formula in Cartesian coordinates. The main advantage of using homogeneous coordinates is that they unify and simplify the mathematics in transformations.

Part 2 - Programming part: You will be programming in C using OpenGL. **You must work on Linux**, and I strongly urge you to work on the lab machines at IC. If you must, you can also work remotely via the matlab server, but be warned that graphics display is **very slow** when working remotely.

Boids

Your task here will be to study the use of simple behavioural rules to animate large numbers of computer generated entities.

As an introduction, watch the following video: <http://www.youtube.com/watch?v=XH-groCeKbE>

Your task for this assignment is to build a simulation of a flock of birds (in computer animation, we call them '**boids**'). Download and uncompress the starter code provided with this assignment into a suitable (empty) directory on your Linux account. Compile the starter code by typing '**./compile.sh**'. The resulting executable sets up a volume in space filled with randomly positioned, non-moving boids.

Your task is to animate these boids, and to provide controls for the user to set the simulation parameters and see how behaviour is affected by changes in these parameters.

A thorough description of your task, as well as references and additional details are provided within the 'Boids.cpp' file. Read the comments and instructions carefully. Implement Everything requested in the sections marked '**TO DO**'

Within the starter code directory, you will find two files called **CHECKLIST** and **REPORT**. You must complete both files. **Failure** to complete these files will result in a mark of **zero** for this part of the assignment. Any descriptions in the REPORT file should be a clear and concise. This is not the place to give technical documentation for your code.

There are multiple parts in the code labeled '**CRUNCHY**'. These are sections where you can do additional work to learn more about this problem, and to earn extra credit. Extra credit will be assessed by your TA in terms of how good your extensions are. Be sure to document any extra Features in the **REPORT**.

Your code will be graded both in terms of functionality and style. It should be properly structured and commented if you want to receive full marks. You can also earn negative marks for bad coding practices and/or style.

Once you're done, compress your working directory into a single compressed **.tgz** file named

boids_studentNo.tgz (e.g. **boids_012345678.tgz**)

This file should contain everything needed to compile/run your code. Submit your work on the CDF server using

submit -c csc418h -a A1 -f boids_studentNo.tgz (if registered in CSC418)

submit -c csc2504h -a A1 -f boids_studentNo.tgz (if registered in CSC2504)

Double check that your compressed file uncompresses properly, and that you can compile and run your solution from it.

B)

1) Does not commute.

Consider a point $(0, 2)$, if we rotate by 90 degrees and translate by $(0, 1)$ we get:

$$(0, 2) \rightarrow (2, 0) \rightarrow (2, 1)$$

But if we do the opposite order we get:

$$(0, 2) \rightarrow (0, 3) \rightarrow (3, 0)$$

2) Does commute.

Rotate X then Y:

$$\begin{bmatrix} \cos X & -\sin X \\ \sin X & \cos X \end{bmatrix} \begin{bmatrix} \cos Y & -\sin Y \\ \sin Y & \cos Y \end{bmatrix}$$

$$= \begin{bmatrix} \cos X \cos Y - \sin X \sin Y & -\cos X \sin Y - \sin X \cos Y \\ \sin X \cos Y + \cos X \sin Y & -\sin X \sin Y + \cos X \cos Y \end{bmatrix}$$

Rotate Y then X:

$$\begin{bmatrix} \cos Y & -\sin Y \\ \sin Y & \cos Y \end{bmatrix} \begin{bmatrix} \cos X & -\sin X \\ \sin X & \cos X \end{bmatrix}$$

$$= \begin{bmatrix} \cos X \cos Y - \sin X \sin Y & -\cos X \sin Y - \sin X \cos Y \\ \sin X \cos Y + \cos X \sin Y & -\sin X \sin Y + \cos X \cos Y \end{bmatrix}$$

3) Does not commute.
 Consider the point $(2,0)$ with
 a translation of $(0,1)$ and a
 reflection on the line $y=x$. If
 we do translation the reflection:

$$(2,0) \rightarrow (2,1) \rightarrow (1,2)$$

If we do the opposite:
 $(2,0) \rightarrow (0,2) \rightarrow (0,3)$

4) Does not commute.
 Consider the vector $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$, if
 we do a scale of 2
 and shear by $\begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}$ we

get:

$$v = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \cdot 2 = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 4 \end{bmatrix} = \begin{bmatrix} 6 \\ 4 \end{bmatrix}$$

If we do the opposite:

$$\begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 5 \\ 2 \end{bmatrix}$$

$$\begin{bmatrix} 5 \\ 2 \end{bmatrix} \cdot 2 = \begin{bmatrix} 10 \\ 4 \end{bmatrix}$$

5) Does not commute.

Consider the point $(0, 2)$ with
a rotation of 90 degrees and
a scale of $(1, 4)$:

$$(0, 2) \rightarrow (2, 0) \rightarrow (2, 0)$$

if we do the opposite:

$$(0, 2) \rightarrow (0, 8) \rightarrow (8, 0)$$