## Maths for Games: Assessment Brief

### Project Brief – Graphical Test Application

#### Overview

This project brief has been designed around the specific, cumulative evidence you must prepare for and present by your assessment milestone to demonstrate you have competency in the requisite knowledge and skills for this subject. Submissions conforming to this brief will provide the specific evidence listed in the table titled *Assessment Tasks and Evidence Description* in the *Assessment Criteria* section above.

You may present additional, or other evidence of competency, but this should be as a result of individual negotiation with your teacher.

#### General Description

For this assessment you are to make a Visual Studio C# project that utilises your Vector and Matrix maths classes. This project will demonstrate the various functions and classes by drawing and manipulating images on screen.

You are to use the RayLib C# 2D framework (available via the resources section for this subject on Canvas: <https://aie.instructure.com/>) when creating your graphical test application.

Once you have set up your project and integrated your maths classes, you will need to draw and manipulate images according to the requirements below.

#### Requirements

Draw a top-down 2D tank, and manipulate its orientation and position using your Vector and Matrix classes in response to keyboard input from the user.

* Draw the tank
  + Rotate the tank by pressing the ‘A’ and ‘D’ keys on the keyboard
  + Move the tank forward or backwards in the direction it is currently facing by pressing the ‘W’ and ‘S’ keys on the keyboard
* Draw the tank’s turret
  + Ensure the turret is parented to the tank such that moving or rotating the tank will also move or rotate the turret (i.e., implement a matrix hierarchy)
  + Rotate the turret independently from the base of the tank by pressing the ‘Q’ and ‘E’ keys
* Fire a bullet
  + Press the space bar to fire a bullet
  + The bullet should spawn at the end of the turret, and travel in the direction the turret is currently facing
  + Ensure the bullet is not parented to either the tank base or the turret (i.e., moving the tank or turret should not move the bullet)
  + Add simple collision detection so that the bullet is destroyed upon collision with the edge of the screen (or optionally another object)

#### Resources

The following Top-Down Tank asset pack may be used, available here: <https://kenney.nl/assets/topdown-tanks>

You are also free to source your own suitable images for this project.

#### Submission

You will need to submit the following:

* A Release build of your application that can execute as a stand-along program
* Your complete Visual Studio project
  + Be sure to remove any temporary build folders (i.e., the bin and obj folders). Only project files, source code files, and any resource files used should be included in your submission.

Package all files in a single compressed archive file (.zip, .7z, or .rar)

#### Submission Checklist

This submission checklist is used to assist your assessor in marking your assessment.

A copy of this checklist can be downloaded from <https://aie.instructure.com/> and must be submitted with your project.

**General**

|  |  |
| --- | --- |
| **Description** | **Y/N** |
| All submitted projects compile without errors *Programs that don’t compile cannot be assessed* |  |
| The program includes a “readme” or document explaining how to compile, execute and operate the program |  |
| The program performs as described in the general description |  |
| The program contains no logical errors |  |
| The code is sufficiently commented and clean |  |
| An attempt has been made to increase the program’s efficiency |  |
| Code compiles without no warnings |  |
| Program executes without crashing |  |
| Program has no memory leaks, and closes all files after use |  |
| A release executable has been made and included in the submission |  |
| Project files and source code are included in the submission |  |
| All files are packaged in a single compressed archive |  |

**Required Features**

Complete the following table by providing the class name or file name, along with the line number, to show where you have implemented each feature.

|  |  |  |
| --- | --- | --- |
| **Feature** | **Class/File** | **Line Number** |
| The tank’s position and orientation are calculated using your Vector and Matrix classes  (any third-party math library included in the framework is not used) |  |  |
| A matrix hierarchy is correctly implemented  (moving the tank base affects the position/orientation of the turret; the bullet is not affected by changes in either the turret or tank base) |  |  |
| The program accepts user input in the manner specified in the requirements above |  |  |

|  |  |
| --- | --- |
| **Feature** | **Y/N** |
| Your Vector and Matrix classes are included in your project |  |
| Your project opens a graphic display window and draws a tank |  |
| The turret rotates correctly |  |
| The bullet spawns at the correct position and travels in the direction the turret is pointing |  |

### Peer-Review Activity

#### Overview

You are required to submit your graphical test application for review by your peers. You are also required to participate in the review of a graphical test application of one or more of your peers. This exercise forms part of your assessment.

You may participate in the peer review while you are still programming your graphical test application, or when upon completion of your application. It is recommended you participate in both peer-review sessions, regardless of what state of completion your application is in.

#### Procedure

The peer-review session will only review the graphical test application.

Take the project for your graphical test application and remove all of your custom Vector and Matrix classes.

You will then give this project (with Vector and Matrix classes removed) to one of your peers, who will test your application using their own maths classes. If you, and they, have written their Vector and Matrix classes to conform to the requirements outlined in the *Assessment Description* section, then the peer-review can proceed.

You should conduct this review together so that any errors that are encountered when integrating different math classes with your application can be solved collaboratively.

If your application cannot be linked with another student’s math classes, this result should be recorded. You should then identify and solve the relevant errors and attempt the review again (either on the same day, or during the next review session).

Once your project is set up and linked correctly, record feedback on the following questions:

|  |  |
| --- | --- |
| Author of project being reviewed: |  |
| Reviewer: |  |
| Date: |  |

|  |  |
| --- | --- |
| Does the code conform to a consistent coding standard?  Note the relevant coding standard and list places where the code can be improved. |  |
| Is the code well commented, easy to read and understand?  List at least one area for improvement or practice you can apply to your own programming. |  |
| Does the program function as intended?  Comment on the mechanics of the application. Note any variation from the brief.  Does the program perform identically on different machines? |  |
| Is the code well structured?  List at least one area for improvement or practice you can apply to your own programming. |  |
| Is vector and matrix math used correctly to draw and manipulation the position and orientation of the game objects?  Note any differences in how calculations are performed between this program and your own. |  |
| Is there anything else noteworthy? |  |
| How would you rate the quality of this project? |  |
| What steps could be taken to resolve any quality issues? |  |

Record the name of the reviewer, along with their responses.

Compile a document (in MSWord or PDF format) that contains all results from all peer review sessions. Also record the names of the people for whom you reviewed code.

Ensure you include a brief outline of any steps you took to resolve any quality issues found in your project.

#### Submission

You will need to submit the following:

* A document in MSWord or PDF format containing the results of the peer review sessions.

#### Submission Checklist

|  |  |
| --- | --- |
| You have participated in at least one peer review session |  |
| The results of all peer review sessions have been recorded (you may use the table above for guidance) |  |
| The name of the review(s) has been recorded, along with their feedback |  |
| You have listed the names of all people for whom you have reviewed code |  |
| The document is neatly typed, with appropriate headings and sub-headings, date, and your name |  |
| Any steps taken to address any quality issues found have been listed |  |

### Number Conversion Exercises

Do the following exercises by hand, without using a calculator (you may use a calculator to check your answers).

Show your working.

1. Convert the following from decimal to binary

* 1
* 42
* 256
* 4,294,967,296

2. Convert the following from binary to decimal

* 10000000
* 10101010
* 11110000
* 11001100

Scan or photograph your work and upload it to the assessment submission section on Canvas for this subject (<https://aie.instructure.com/>).