

MS1 – Hybrid Engine {2D, 3D, Animation}

Student Information

Integrity Policy: All university integrity and class syllabus policies have been followed. I have neither given, nor received, nor have I tolerated others' use of unauthorized aid.

I understand and followed these policies: Yes No

Name:

Date:

Submission Details

Final **Changelist** number:

Verified build: Yes No

YouTube Link:

Required Configurations:

Discussion (What did you learn):

YouTube Process

- Record the YouTube demo
 - You need to record with commentary
 - Suggestion: **Filmora** or **OBS** screen capture
- Record the desktop (enough to show your directory and the visual studio and output)
 - Show your directory in recording
 - Launch the visual studio (double click solution)
 - Show off relevant parts of the code with commentary
 - Launch and demo the milestone
 - Watch your video
 - Verify that video clear and can you hear the commentary with audio in stereo?
- Note: Milestones Demo cannot be longer that 5:00 mins
 - If you go over... do it again
- Publish your YouTube recording
 - Make sure it is accessible without any login or permission to play
 - It can be private but not restrictive to play by anyone with the link
- Submit your code to perforce to the appropriate PA directory
 - Verify it
- Submit your YouTube shortcut to student directory:
 - **MS1_FirstName_LastName.url**
 - Example: MS1_Ed_Keenan.url
- Place the shortcut in your student root directory
 - //ClassDepot/student/**CampusConnectID/MS1_FirstName_LastName.url**
 - **Example:**
 - //ClassDepot/student/**ekeen2/MS1_Ed_Keenan.url**

Pdf form (this document)

- *Submit this PDF to perforce*
 - *Fill in form*
 - *Name, changelist, etc...*
 - *Submit back to perforce*
 - *Check it out*
 - *Submit it back to perforce to the same location*

Verify Builds

- Follow the Piazza procedure on submission
 - Verify your submission compiles and works at the changelist number.
- Verify that only MINIMUM files are submitted
 - No – Generated files
 - *.pdb, *.suo, *.sdf, *.user, *.obj, *.exe, *.log, *.pdb, *.db, *.user
 - Anything that is generated by the compiler should not be included
 - No – Generated directories
 - /Debug, /Release, /Log, /ipch, /.vs
- Typical files project files that are required
 - *.sln, *.cpp, *.h
 - *.vcxproj, *.vcxproj.filters, CleanMe.bat

Standard Rules

Submit multiple times to Perforce

- Submit your work as you go to perforce several times (at least 5)
 - As soon as you get something working, submit to perforce
 - Have reasonable check-in comments
 - Points will be deducted if minimum is not reached

Write all programs in cross-platform C++

- Optimize for execution speed and robustness
- Working code doesn't mean full credit

Submission Report

- Fill out the submission Report
 - No report, no grade

Code and project needs to compile and run

- Make sure that your program compiles and runs
 - Warning level ALL ...
 - NO Warnings or ERRORS
 - Your code should be squeaky clean.
 - Code needs to work "as-is".
 - No modifications to files or deleting files necessary to compile or run.
 - All your code must compile from perforce with no modifications.
 - Otherwise it's a 0, no exceptions

Project needs to run to completion

- If it crashes for any reason...
 - It will not be graded and you get a 0

No Containers

- NO STL allowed {Vector, Lists, Sets, etc...}
 - No automatic containers or arrays
 - You need to do this the old fashion way - **YOU EARNED IT**

Leave Project Settings

- Do NOT change the project or warning level
 - Any changing of level or suppression of warnings is an integrity issue

Simple C++

- No modern C++
 - No Lambdas, Autos, templates, etc...
 - No Boost
- NO Streams
 - Used fopen, fread, fwrite...
- No code in MACROS
 - Code needs to be in cpp files to see and debug it easy
- **Exception:**
 - implicit problem needs templates

Leaking Memory

- If the program leaks memory
 - There is a deduction of 20% of grade
- If a class creates an object using new/malloc
 - It is responsible for its deletion
- Any **MEMORY** dynamically allocated that isn't freed up is **LEAKING**
 - Leaking is **HORRIBLE**, so you lose points

No Debug code or files disabled

- Make sure the program is returned to the original state
 - If you added debug code, please return to original state
- If you disabled file, you need to re-enable the files
 - All files must be active to get credit.
 - Better to lose points for unit tests than to disable and lose all points

Allowed to Add files to this project

- ~~This project will work "as is" do not add files...~~

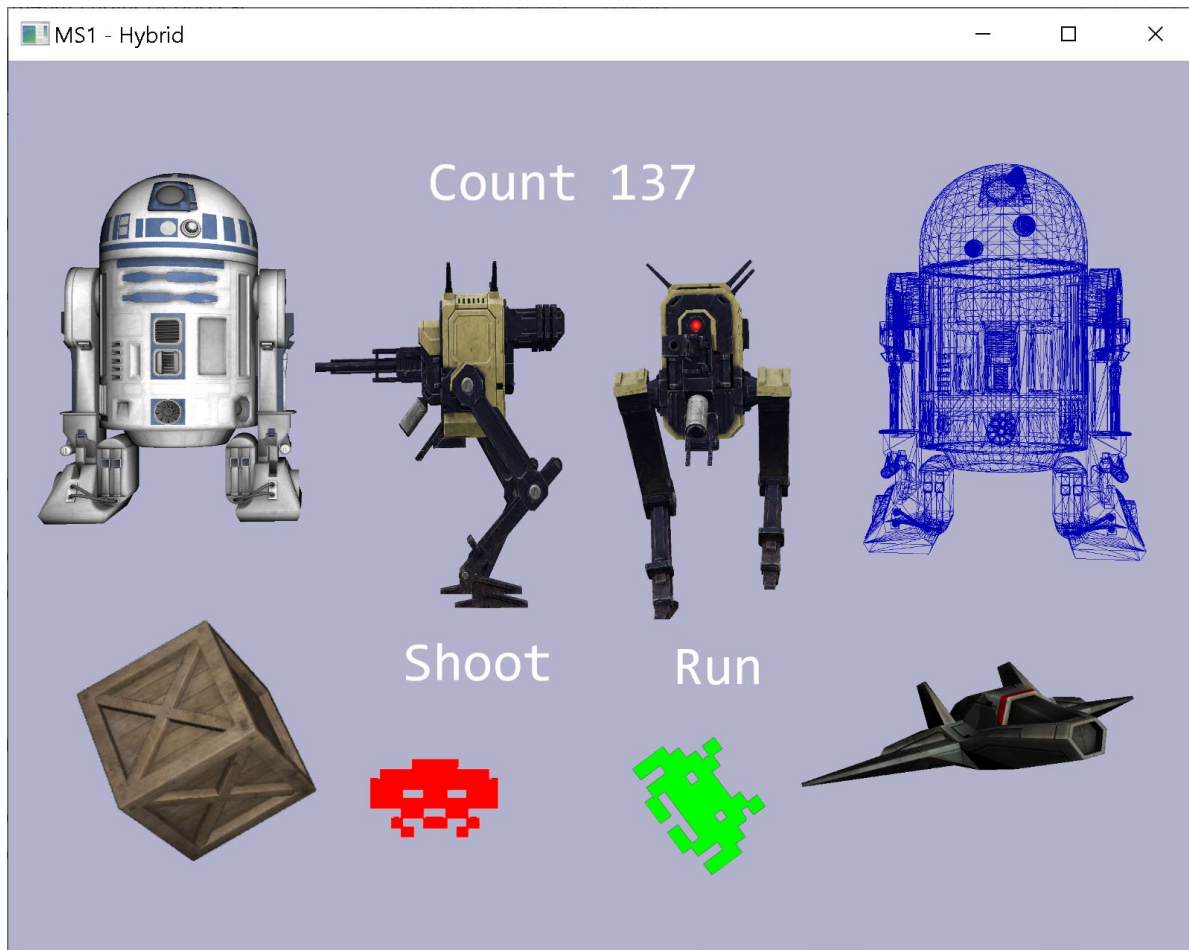
Due Dates

- See Piazza for due dates
- Submit program performe in your student directory assignment supplied.
- Fill out your this **Submission Report** , **PA** and **YouTube link** to perforce
 - **ONLY** use Adobe Reader to fill out form, all others will be rejected.
 - Fill out the form and discussion for full credit.

Goals

- Learn
 - Protocol Buffers for all assets
 - Texture, 3D Meshes, 3D animation Clips, Fonts
 - 2D Game Engine with Sprites
 - 2D Sprites
 - Font systems
 - 3D Engine with at least 2 models being displayed
 - 3D Animation with support
 - 2 animations on the screen playing different animation clips
 - No Memory Leaks!!

Assignments



Mockup of a screen.

1. Export at *least* 4 Animations from 1 skeletons

- Displaying two different animation clips on the screen at the same time
- You **NEED** to show animations switching **Dynamically**
 - Use a keyboard press to change animations
 - Switching animations
 - Does not need to be blended with other animation
 - A hard switch is OK
- You **NEED** to show two different skeletons at the same time
 - Example: ChickenBot – run and shooting

2. At least 2 different 3D rigid models need to be displayed

- These models should rotate to show the matrices are working

- At **DIFFERENT** rates of rotation
 - You can display more than 2 models...
 - At least two need to be different
 - With different rotations
- 3. Demonstrate the 2D font system**
- Place 2-3 strings on the screen
 - One string needs to be dynamically changing
 - Example: counting
- 4. Demonstrate the 2D Sprites**
- Place 2-3 sprites on the screen
 - Aliens red and green for example
- 5. All data needs to be in Google Proto buffer**
- Texture, Animation Skeleton, Animation Data(clips)
 - 3D Models, Font data
- 6. No MEMORY leaks**
- Make sure your project isn't leaking at All
 - Run in Debug with the new Memory tracking enabled

Validation

Simple checklist to make sure that everything is submitted correctly

- Submitted project to perform correctly
 - Is the project compiling and running without any errors or warnings?
 - Is the submission report filled in and submitted to perform?
 - Follow the verification process for perform
 - Is all the code there and compiles "as-is"?
 - No extra files
 - Is the project leaking memory?
- Submitted the YouTube link to perform?

Hints

- Do this assignment by iterating and slowly growing your project
 - You won't be able to finish this assignment in one day - Start now
- Getting the 3D, Animation and 2D working together
 - They need to be working first before you can tweak layout and demo
- 3D game object and Animation object should be cut and paste from GAM 575
 - But now its needs to be in Google Protocol Buffers
- 2D is the new part...
 - Follow the demo in class
 - 2D Sprites
 - Fonts
- Get one sprite working
 - Then focus on the font system
- Memory Leaking
 - Use the new memory system to find your leaks