CSGI.GA.2270 – Computer Graphics

Fall 2022 Assignments

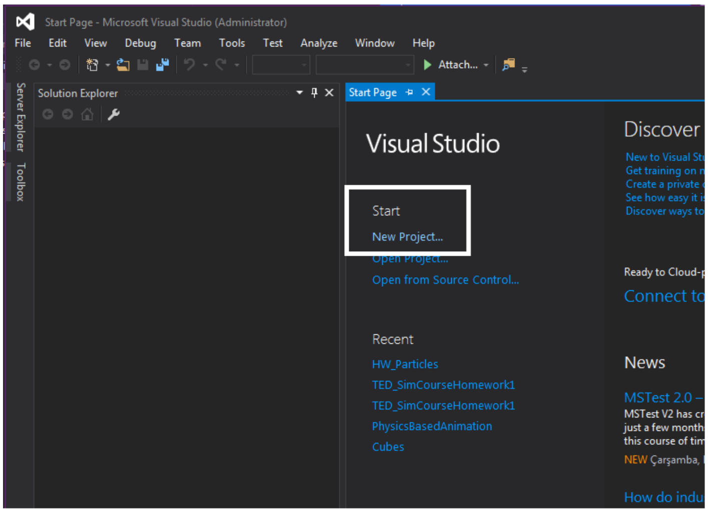
Simulation Assignment

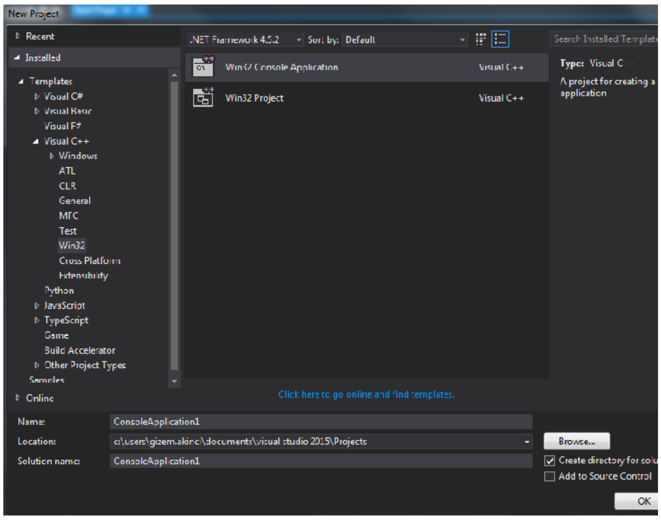
Due date: 12.07.2022, 11:59pm

The aim of this assignment is to make you get familiar with simple simulation systems. For the scene hierarchy, you will get help from freeglut. glut was superseded by freeglut and it is still used for smaller projects, but I wouldn’t recommend it if you needmore than a quick demo.

OPENGL SETUP

- This setup is for Visual Studio on Windows OS. You can use a different IDE and OS but a helper doc will not be provided for them.
- Download shared files
- Create a new folder, e.g. GL_Folder, to put the content (glew and freeglut)
- Create an empty C++ project in Visual studio (see below).





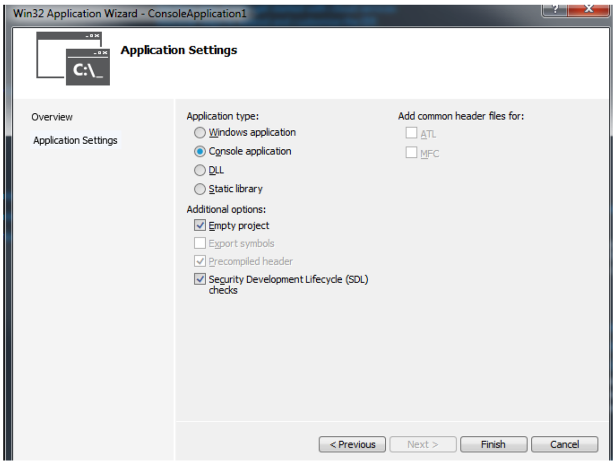
Name: ConsoleApplication1

Location: c:\users\gizem.akinci\documents\visual studio 2015\Projects

Solution name: ConsoleApplication1

☑ Create directory for solu

□ Add to Source Control



Application type:

○ Windows application

● Console application

○ DLL

○ Static library

Additional options:

☑ Empty project

□ Export symbols

☑ Precompiled header

☑ Security Development Lifecycle (SDL) checks

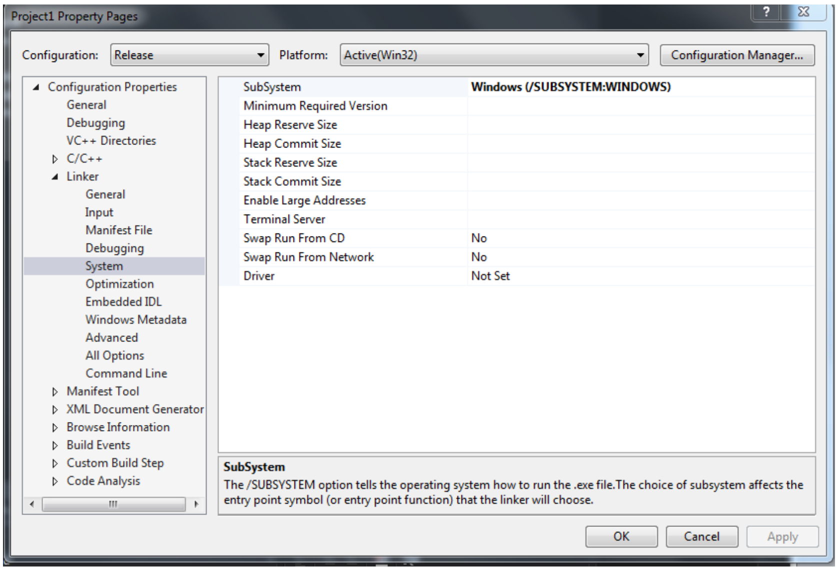
Add common header files for:

□ ATL

□ MFC

- Right click Project → Properties →Configuration Properties → VC++ Directories →Include Directories:
  Select ***Include*** folders of both glew and freeglut
- Configuration Properties → Linker → Input:
  Additional Dependencies:
  Write: ***glew32.lib*** and ***freeglut.lib***
- Configuration Properties → Linker → General:
  Additional Library Directories:
  Select ***lib*** folders of both glew and freeglut
- Copy ***glew32.dll*** and ***freeglut.dll*** from GL_Folder and paste them to your project folder

Be sure that your subsystem is Windows:

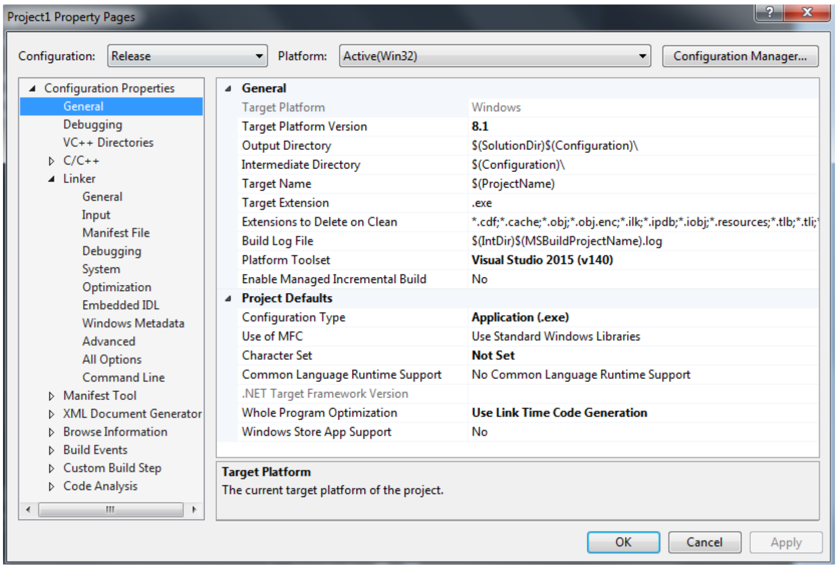


Configuration: Release
Platform: Active(Win32)

SubSystem	**Windows (/SUBSYSTEM:WINDOWS)**
Minimum Required Version	
Heap Reserve Size	
Heap Commit Size	
Stack Reserve Size	
Stack Commit Size	
Enable Large Addresses	
Terminal Server	
Swap Run From CD	No
Swap Run From Network	No
Driver	Not Set

SubSystem
The /SUBSYSTEM option tells the operating system how to run the .exe file.The choice of subsystem affects the entry point symbol (or entry point function) that the linker will choose.

And your character set is set to: Not Set



Configuration: Release
Platform: Active(Win32)

General	
Target Platform	Windows
Target Platform Version	**8.1**
Output Directory	$(SolutionDir)$(Configuration)\
Intermediate Directory	$(Configuration)\
Target Name	$(ProjectName)
Target Extension	.exe
Extensions to Delete on Clean	*.cdf;*.cache;*.obj;*.obj.enc;*.ilk;*.ipdb;*.iobj;*.resources;*.tlb;*.tli;
Build Log File	$(IntDir)$(MSBuildProjectName).log
Platform Toolset	**Visual Studio 2015 (v140)**
Enable Managed Incremental Build	No
Project Defaults	
Configuration Type	**Application (.exe)**
Use of MFC	Use Standard Windows Libraries
Character Set	**Not Set**
Common Language Runtime Support	No Common Language Runtime Support
.NET Target Framework Version	
Whole Program Optimization	**Use Link Time Code Generation**
Windows Store App Support	No

Target Platform
The current target platform of the project.

PART I: PARTICLES – NUMERICAL INTEGRATION SCHEMES

In this part, our ourpose is to create a simple particle system. Please check the empty fields in the given code which are commented with //TO DO line.

TASK 1 – INTEGRATION SCHEMES

In this part, you are expected to implement Euler, Euler Cromer and Verlet methods. The first thing you need is taking the current value of position and velocity, then update them according to corresponding scheme and set the position and velocity of the particle with new values. Remember:

Euler:

$$\mathbf{x}_{t+h} = \mathbf{x}_t + h\mathbf{v}_t$$
$$\mathbf{v}_{t+h} = \mathbf{v}_t + h\frac{1}{m}\,\mathbf{F}_t$$

Euler-Cromer:

$$\mathbf{v}_{t+h} = \mathbf{v}_t + h\frac{1}{m}\,\mathbf{F}_t$$
$$\mathbf{x}_{t+h} = \mathbf{x}_t + h\mathbf{v}_{t+h}$$

Verlet:

$$\mathbf{x}_{t+h} = \mathbf{2x}_t - \mathbf{x}_{t-h} + h^2\frac{\mathbf{F}_t}{m} + O(h^4)$$

Velocity independent
if needed: $\frac{\mathbf{x}_{t+h}-\mathbf{x}_t}{h} = \mathbf{v}_{t+h}$

TAKS 2 – ENERGY COMPUTATION

In the second part, you are expected to compute system energies. So, traverse along the particle list, and update kinetic and potential energy accordingly.

Hint: for potential energy, *h* is computed as: particlePosition.y + halfWorld.getY().

TASK 3 – COMPARISON

Try your system with different integration schemes and with different number of particles. Does energy computation result give an idea about the schemes' stability? Discuss.

PART II: RIGID BODY CUBE

In this part, our purpose is to simulate a simple rigid body cube (see Figure 1). Please check the empty fields in the given code which are commented with //TO DO line.

TASK 1 – FORCES

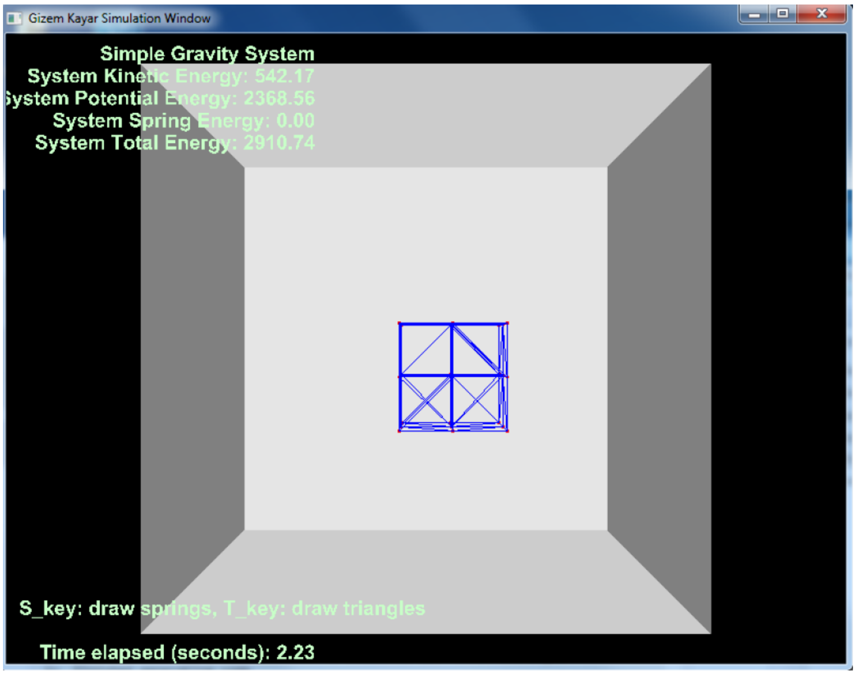
In this part, you are expected to apply forces to particles and springs appropriately (in Simulation.h). First of all, traverse along the particles and apply gravity to all of them. Later, traverse along the spring list and apply spring and damping forces. Please check Hooke's Law and damping forces.

TAKS 2 – ENERGY COMPUTATION

In the second part of the homework, you are expected to compute system energies (in Simulation.cpp). First of all, traverse along the particle list, update kinetic and potential energy accordingly which is similar to your first programming assignment. Later, traverse along the spring list and compute total spring energy of the system. Reminder: **F** = k**x**, and thus $\mathrm{E} = \frac{1}{2}k\boldsymbol{x}^2$

TASK 3 – COMPARISON

Try your system with different integration schemes, increase/decrease spring and damping constants (in Simulation.h). What kind of behavior you observed with different setup? Discuss the results.



```
       Simple Gravity System
   System Kinetic Energy: 542.17
System Potential Energy: 2368.56
    System Spring Energy: 0.00
    System Total Energy: 2910.74

S_key: draw springs, T_key: draw triangles

  Time elapsed (seconds): 2.23
```

Figure 1: Simple Rigid Body Simulation