# Game Overview:

The team’s proposed game is the creation of a “museum” like environment where they can showcase different elements/requirements of the assessment in the environment. This can be done in either separate rooms or a single larger room. The environment will be used to demonstrate the following:

* Camera:
  + The game will be played from a first person perspective, similar to a First Person Shooter.
  + The camera is attached to the player head(or where the head would be)
* AI:
  + “Visitors”, who are wandering characters with random/specific paths and will try to avoid blockers on their path
  + A following entity, possibly for the resemblance of “a dog” that follows the player
  + An entity that on player collision will promptly apologise and go on a new walking path
* Exhibits:
  + Will be used to demonstrate different areas of study: lighting, shadows, collisions(also available in other parts of the game) and/or different kinds of shaders
* Collisions:
  + Collisions will be available through the environment
  + These will be used sparingly, avoiding their use in areas where players shouldn’t be able to reach
* Interaction:
  + The player will be able to interact with various exhibits in the museum
* Graphics:
  + There will be entities with different models in the environment (exhibits, objects, entities, etc.)
  + Different types of shaders will be used, from more simpler ones such as Pixel/Vertex shaders to possibly more advanced ones made by the team
  + The environment will feature lighting and shadows

This leaves the project open to possible additions if the time allows it. These can be:

* Sound, both for the environment and its entities
* Procedural generation of rooms
* Animations

# Game Breakdown:



The game was split into reasonable sized chunks and elements deriving from each other. Considering previous experience (Computer Games Assignment – Develop a 2D Game) the team has decided that the graphical side (which is also a more DirectX intensive side) will require more work and time to proper develop.

Considering the above, the “Logic” side of the game, while covering more aspects of a functional game, may require less or equal amounts of time and effort comparatively to the “Graphics” side.

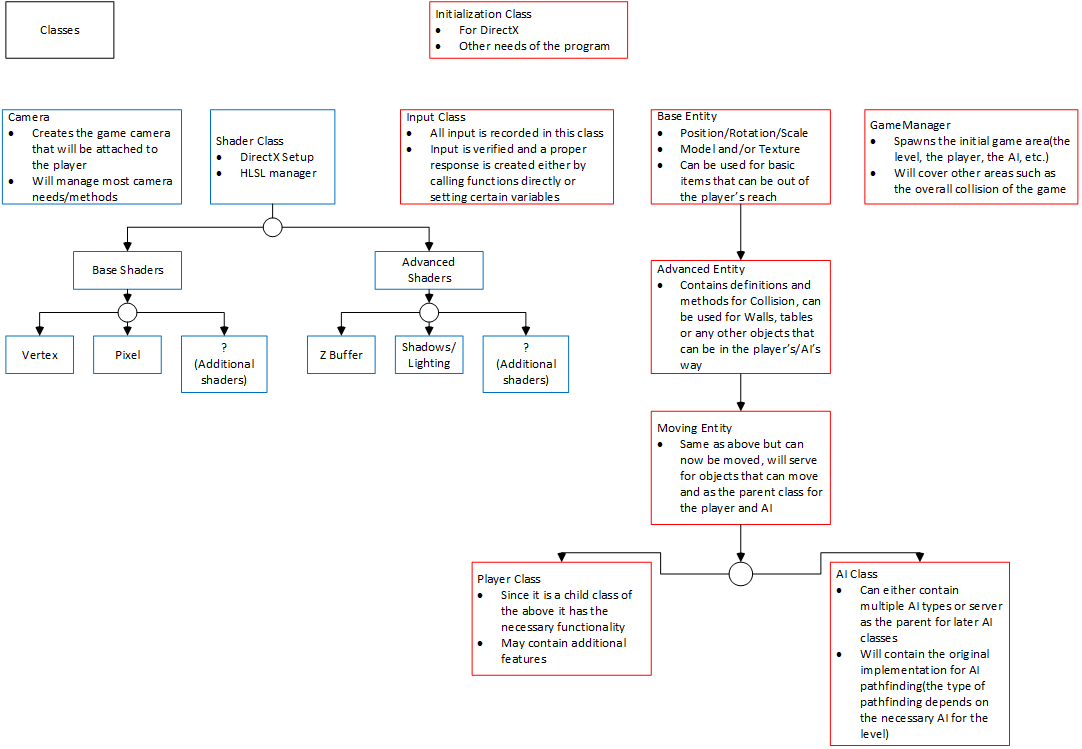
A single programmer cannot cover multiple areas of study, especially on big projects. We already see in the industry that there are different areas of specialisation: gameplay programmers, AI programmers, graphics engine programmer, sound programmer, etc.

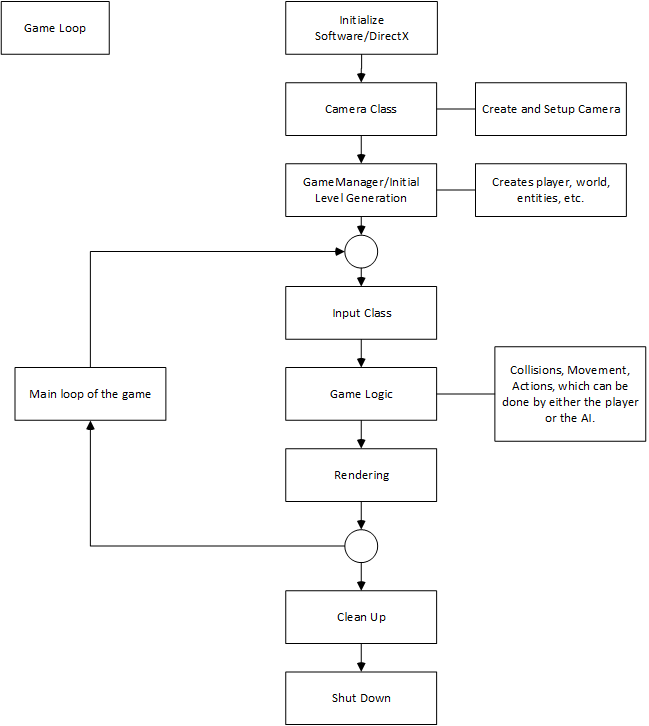
Using the above information as a basis the team has agreed to split the work as follows:

* George Alexandru Ciobanita will take the “Logic” part described above
* Nedelin Gochev will take the “Graphics” part described above

This way, members can focus on specific areas of study and research, without fear of confusion from covering too many areas. Thus members can provide proper information to each other and communicate new findings in a better manner.

# Basic Class Diagrams & Game Flow:





There will be final updated version of the above as the team approaches the hand in date. These version will contain updated information from the resources gathered.

# Version Control & Log Keeping/Testing:

The following are proposed for better and safer work conduct:

1. Github, in order to keep files, versions and backup available online.
   1. Team members will upload files and information, with appropriate descriptions and titles, as they acquire it and keep it available in the “AGP-Assignment” repository.
   2. Members can create/update files as they see fit (there is a document available in the repository that each member has to update).
   3. Members will communicate with each other in order to clearly update any current piece of information. There is the possibility that the findings of one member can help the other realise mistakes or improve.
2. Visual Studio can be setup to create logs after the codebase has been compiled and tested.
3. For future use the team can employ the use of “assert” a function mainly used for testing. This can work well with the Visual Studio log implementation as the following are logged: the function expression, name of source file, and the line number where it happened (e.g. Assertion failed: *expression*, file *filename*, *line number*).
4. Grey-box testing, a combination between White & Black box testing, can be used through implementation.
5. The tutorials available from Advanced Games Programming give a good view in how to setup a project in order to properly get error/warning messages when certain systems do not function. This can be specifically seen in Tutorial 01 and 02.