Programming 4 – Exam Assignment

As exam assignment this year you will be making a game engine on your own and create a well-known game with it – Pac Man.

Game requirements

- The player can walk around on a map as Pac Man, gaining points by eating yellow pill-like thingies on the ground.
- Meanwhile a bunch of ghosts try to locate him and catch him.
- Special pills make the ghosts scared and Pac Man can eat them too for a while
- Pac Man has three lives.
- There is only one level.
- It has keyboard and controller support.
- There is a single player mode.
- There is a two-player mode. The second player can choose to be Mrs Pac Man or a ghost.

Game engine requirements

- Use Minigin as a start point for your game engine.
- Make sure the project builds for all build targets at warning level 4 and warnings flagged as errors.
- Apply all best practices we've seen
- Apply software design patterns and game programming patterns as you see fit, but we expect to see at least Game Loop, Update method, Command and Component.
- Use threads to run the ghosts

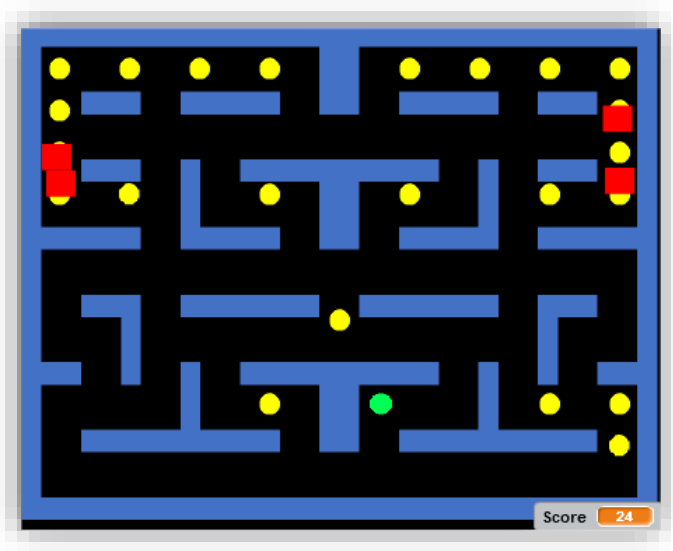
Optional nice to have's

- Level file loading support (recommended)
- Multiple levels
- Sound

- Fruit in the level that gives extra points

Recommendations

Don't spent too much time on the visuals. After all, we're programmers and not artists so we don't care one bit about the look of the game, just make it work. This design for example would suffice:



Use a source control provider. We recommend using Github or Bitbucket. SourceTree is a recommended tool to work with these.

Use at least Visual Studio 15.5.6

Play fair – don't copy code from your colleagues. Submissions will be run through plagiarism detection software.

Deliverables

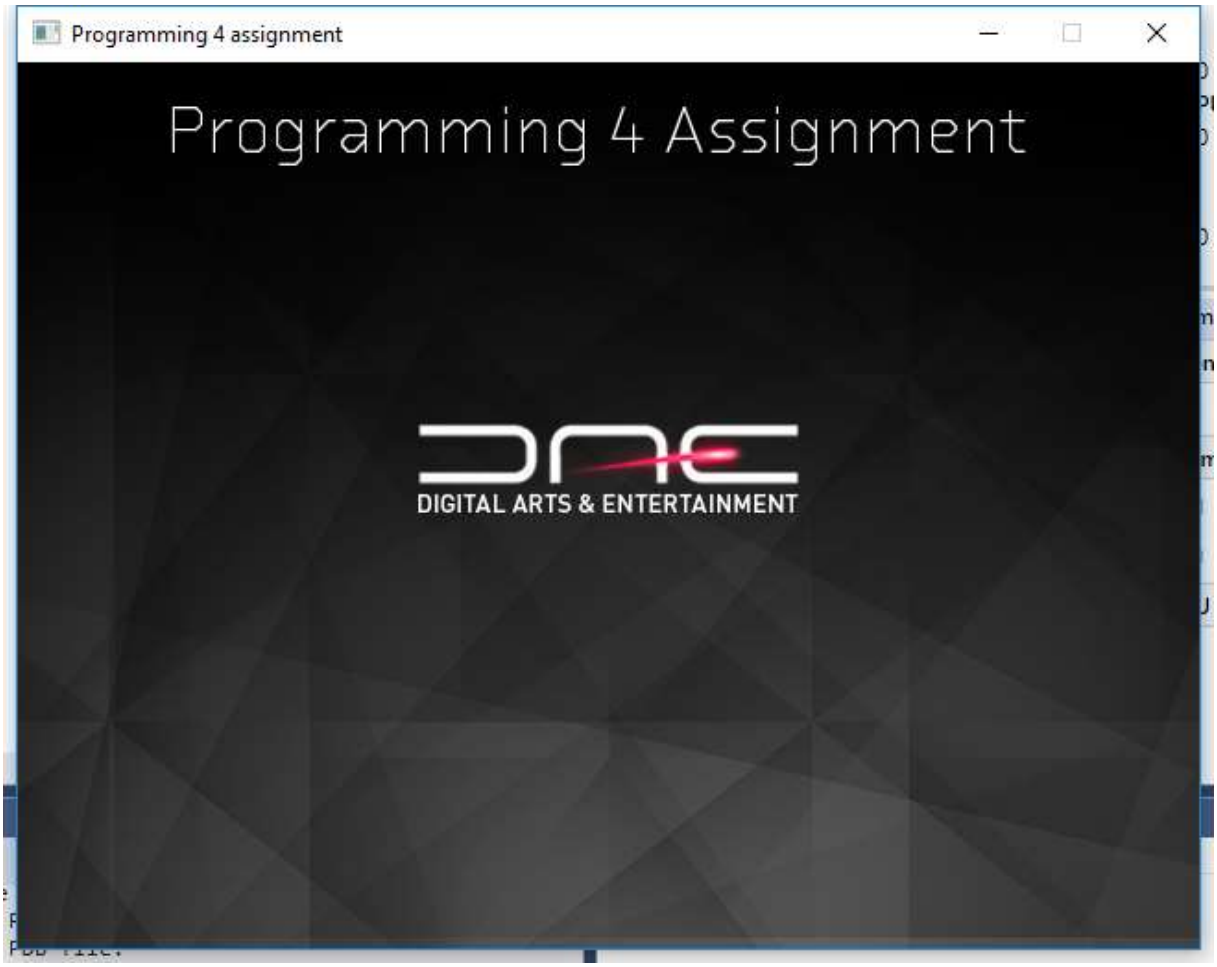
As deliverable we expect a zip file containing

- The source code of your project including one visual studio solution and project file(s).
- All build targets must compile and deliver a working executable.
- An x86 and an x64 Release build
- A Readme.txt containing
    - Some specifics about your engine and the design choices you made
    - A link to the source control depot you used for your project

This project will weigh 60% on your total score.

About Minigin

Minigin is an SDL-based project build specifically to get you started.



This project links with

- SDL 2.0.7 (https://www.libsdl.org/)
- SDL_image 2.0.2 (http://www.libsdl.org/projects/SDL_image/) for png and jpg support
- SDL_ttf 2.0.14 (http://www.libsdl.org/projects/SDL_ttf/) for text rendering support
- GLM 0.9.8.5 (https://glm.g-truc.net/0.9.8/index.html) as math library (header only)

This project is a starting point, meaning that you are free to change anything you like. You should, because some of the classes aren't the best design choice in a game engine. Feel free to add/remove libraries.

We'll highlight a few files/classes:

Minigin.cpp

Contains the main function. Has an Initialize and Cleanup function which should be sufficient. Has a LoadGame function that you'll probably want to change. Contains a rudimentary game loop.

Texture2D and Font

Are RAII wrappers for SDL_Texture and SDL_Font respectively.

InputManager
Is the same class we've seen in the weekly assignment of Week 4. Adds support for controllers.

Log.h
Is a convenience header – it makes sure that all output written to std::cout, which normally goes to the console window, is now written to the Debug Output window in Visual Studio.

Singleton.h
Convenience implementation of a singleton template base class. Remember what was discussed in week 4 before using it.

SceneObject
Interface for objects in the scene

GameObject
General GameObject in the scene. Has a transform and a texture.

TextObject
Object to place text in the scene. Has a transform and a string. Renders to a texture.

Transform
General class for positional data in the scene. Only supports positions for now.

ResourceManager
Used to load textures and fonts from the Data folder of the project. Accepts paths relative to a given Data folder.

SceneManager and Scene
Scene containing the SceneObjects of the game. The SceneManager contains the scenes in the game.

Hints
Implement step by step. Don't try to add everything at once, work in pieces and submit regularly to your source control.

For example: start by implementing a scene graph with components.

Questions?
Don't hesitate to mail us:

alex.vanden.abeele@howest.be

tom.tesch@howest.be

Or ask in person.