CS3021 Intermediate Programming Practical Application (Big) Project

The course practical application project is an self-directed exploration and implementation of a project or your choosing. You may generate a project idea from scratch or chose to implement an old school arcade-style game. The Pygame feamework is the arcade game default. The instructor is not a Pygame expert, only count on big picture help there. This should tie together many of the individual topics covered in the leacture/lab part of class and provide substantially more opportunity to consolidate the topics into lasting programming skill.

There are Two Milestones

- Milestone 1: Form your team and choose your project topic. Deliverable: Free form propject proposal and team list. Following the proposal submission we will have a short 5-10 minute chat to set up general scoping ideas and prepare you for executing milestone 2.
- Milestone 2: Scope and project implementation plan document. Scope discussion indicating desired feature inclusion, this will form the minimum requirements for implementation. Initial project implementation plan covering overall project organization and code outline, includes description broken down into code modules, identification of data organization/structures and initial planded class needs. Class descriptions should map to the TYPE description referreed to in class. Deliverable: Free form discussion of what scope of effort entails, preferrably containing a list of required and stretch features; code organization documentation.

Final Submission

Completed projects due NLT final day of finals period. 5 point bonus for projects demonstrated live in-class and another 5 if submitted by completion of last regularly scheduled class period.
Here is my grading breakdown.

Deliverable: A singe zip posed to Sakai which includes:

1. **Project code directory**
2. **Documentaition directory** 1) Basic documentaion and 2) A technique callout sheet.

   1) The basic documentaion to allow a user to run on their machine and understand how to operate, this includes things like dependencies or specific procedures. I am not looking for in-depth "game procedures or rules", just enough so that the GUI can provide its role of showing where to go once it presents itself. Also ensure this eliminates "Secret Sauce" problems, where it runs on your dev machine but nowhere else until the secret sauce is applied. Even if you do this in one file, place it in a dirctory titled Documentation. Remember to test on a Mac and ensure any requirements I will need to run on my machine are covered, it is enough to say Pygams installed for that requirement unless you also require extensions.

   2) A cross reference of the required and optional techniques to filename/line number where that technique is implemented in your project. Just one or two examples each need to be cross references, not every appearance of a technique.
3. **Reactions directory** A paragraph (max two) from each team member summarizing insights gained and lessons learned from the project process. Not only concentrating on obvious successes, but also noting

challenges which you overcame and resultingly will have an impact on how you attack problems in the future.

Project Technique Inclusion Guidelines

I will be looking for these techniques displayed in the implementation of your project. Identify at least one instance of each in your technique callout sheet.

1. Basic object functionality as covered in class.
2. Exception handling.
3. Basic Inheritance (the geometry classes as provided do not count towards this.)
4. GUI use including buttons and a text input functionality.
5. Functional programming, minimum of use of function name used as an argument.
6. Unit tests for non-GUI functions. This means functions that directly interact with GUI events do not need unit tests (they best use a technique called functional testing), functions that indirectly interact with GUI functionality do need unit tests.
7. Use of a Framework or Library not covered in class (Pygame counts towards this)
8. Use of at least two data structures (assessed as part of code quality, call out best two)

Optional techniques/functionality. As applicable, identify at least one instance of each in your technique callout sheet. Up to a 10 point bonus at my discretion. Max bonus entails significant and clever use of multiple techniques.

1. Significant functional programming use, more in depth than just button functions/lambdas.
2. Grab bag topic techniques
3. Use of a Framework or Library not covered in class (Pygame counts towards this)
4. Some form of data export and/or persistence, such as JSON or pickling
5. Other clever and useful techniques not covered in class which you learned and used efectively.

Choosing a Topic -- A Default Approach:

Implement a game utilizing the Pygame library. Old school games such as those with bitmapped graphics are targeted. Examples are Space Invaders, Breakout and Tetris. The list is illustrative, not restrictive. More team members should equate to more/richer feature inclusion plans. Typical features would include things like basic gameplay with keyboard i/o, a level transition, high score board for starters. Use of the class generated geometry modules is a good place to start thinking about game engine design and implementation.