# Grading Elements

## Game Features

* Entities:
  1. Blaster
     + Basic sounds, movement & death
     + Single bullet at a time
     + NPC and Keyboard controls
  2. Mushroom
     + sounds
     + 4 lives
     + Poison states & regeneration
  3. Centipede
     + Basic sounds, movement & death
     + Smooth 180 turns at mushrooms, ignores mushrooms during a turn
     + Stays in the bottoms rows
     + Splits when segment hit
     + Drops to the bottom and head hits poisoned mushroom
  4. Scorpion
     + Basic sounds, movement & death
     + Poisons mushrooms
  5. Flea
     + Basic sounds, movement & death
     + Two lives, drops faster after one hit
  6. Spider
     + Basic sounds, movement & death
     + Score on death based on distance (and shown)
* Game Modes:
  1. Demo mode
     + Display high scores
     + Game runs normally (characters block centipede), but all sounds off
  2. Game Mode
     + 1 or 2 players, 3 lives each
     + Game proceeds in waves, each with various specs
     + Players alternate, each on its own wave
     + On death, if high score reached, player asked to enter 3 letter initials
* Scoring:
  1. Scoring points implemented as specified in the info page of the game
  2. HUD displays score/lives/high score correctly

## Code Elements

Some expected coding elements:

1. Use of Strategy Pattern:
   * Blaster controls (keyboard vs. NPC)
   * Others?
2. Use of Observer pattern throughout the code (too numerous to list)
3. Use of Simple Factory pattern for game entities,
   * Includes recycling of instances
   * Keeps track of active elements (used for clearing a level)
4. Use of Singleton pattern where appropriate
   * Factories, managers and possibly a few others
5. Centipede
   * Centipede head & segment are subclassed from a base class
   * Linked movement
6. Mushroom field housekeeping
   * References to mushrooms
   * Various counters and lists to minimize work
7. Wave control
   * Wave data specified in human-readable text file, read once only
   * Monitors the game and triggers the appropriate events
8. Player
   * Combines the state of the mushroom field, score, life, wave, etc.
9. Text display
   * Characters implemented as Flyweight pattern
   * Characters should have minimal processing when not changed
10. Sound manager
    * Used to control silent/sounds based on game mode
11. High scores
    * Stored in text file, read once only.
    * Update file only when new high score entered

## Code Practices

This is not meant to be an exhaustive list, but more a reminder of common errors/omission/sloppiness

* The code should be designed to minimize work at all times
  + Track data as it changes rather than recomputed it
  + Use callbacks rather than repeated testing
* Avoid code duplication
  + Use of Class/subclass
  + Functions for repeated operations
* Functions block should be small and manageable
* Class/Function/Variable names should be meaningful (and properly spelled!)
* Comments to identify purpose of method/variable where appropriate
* In-code comments to clarify less obvious steps (note: breaking up the operations into more obvious steps might be preferable)
* Never use raw values: define clearly identified constants instead
* Use proper indentation at all times. Spaces are preferable over tabs

## Grading Rubric

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Categories** | **Levels of completion** | | | | |
| **5pts**  **Participation**  **(Class & Forum)** | 5pts   * Very active * Provides much help/support to classmates | 4pts   * Active * Helped or requested help during code reviews | 3pts   * Active if prompted * Collaborated outside class | 2pts   * Active if prompted * No interest in code review | 1pt   * Barely present |
| **20pts**  **Communication &**  **Project Planning** | 20pts   * Always on track * Clear reports | 16pts   * 1-2 v-, no 0 * Clear reports with explanations | 12pt   * 3-5 v-, no 0 * Clear reports with explanations | 8pt   * 3-5 v- or 0 * Clear reports with explanations | 4pt   * >5 v- or 0 * Unclear reports |
| **40pts**  **Game Features**  **(Implemented & functional)** | 40pts   * All entities * Scoring and HUD * Both game modes * Wave control * High score system | 32pts   * All entities * 3 of the others | 24pts   * All entities * 2 of the others | 16pts   * Entities only * 1 of the others | 8pts   * Most entities |
| **20pts**  **Coding Elements**  **(Implemented & functional)** | 20pts  All code elements implemented using the best design pattern and as efficient as possible | 16pts  All 5 design patterns used, but some elements not implemented efficiently | 12pts  4 of the 5 patterns used. Code elements mostly efficient | 8pts  Only 3 of the patterns used. Efficiency problems | 4pts  Only 1 or 2 pattern used. Efficiency problems |
| **10pts**  **Code Practices** | 10pts  All code practices followed | 8pts  Occasional lapse | 6pts  Some recurring problem | 4pts  Many lapses and recurring problems | 2pts  It’s mostly a mess… |
| **5pts**  **Post-Mortem Discussion**  **and Assessment** | Grade based on answers to the final questionnaire (essay-style answers) | | | | |