1. **User-Interface (implements GLSurfaceView for OpenGL Support):**
   1. Set up initial screen/menu/etc.
   2. Create and start the Input Service
   3. Create an empty Game object into a new Thread
   4. Wait for player to select “New Game” from menu
      1. Now begin handling New Game processes
2. **New-Game/Restart-Game processes:**
   1. Initialize Game object
      1. Create a Looper, and attach it to the Main UI Looper via getMainLooper();
      2. Create an empty View object (implements Runnable)
      3. Create and/or initialize Snake object and randomize starting position/direction
      4. Create and/or initialize Terrain
         1. Randomly generate terrain obstacles and list of power-ups
         2. Place the Snake onto the terrain
      5. Initialize the Game Timer
      6. Initialize the Player’s Score
   2. **Begin Game:**
      1. Every x-milliseconds:
         1. Request the Input data from the Input Service
            1. Move the snake in the direction of the last-known Input

Check to see if the head of the snake is currently over-lapping it’s body

If so, game-over

If not, game continues

Check to see if the head of the snake is currently over-lapping a power-up

If so, apply power-up

If not, do nothing

Check to see if the head of the snake is currently over-lapping an obstacle

If so, determine the best direction to turn the snake

If no potential alternative direction exists besides a 180-degree rotation, then game-over

Else, turn snake

If not, do nothing

* + - * 1. Update View

Construct a Bitmap using:

Terrain Object (Obstacles and their shapes/positions)

Snake Object (Position, Shape, Texture, etc.)

Power-ups (Sprites, Positions, Types, etc.)

Update Bitmap of View object

* + - * 1. Display View

Post View Object (Runnable) to UI-Looper Queue

* 1. **Game Over:**
     1. Display Player’s Score
     2. Wait for either new game or end game
  2. **Collision Detection:**
     1. Progressive-Scan or Binary Bit-Mask (can’t put the head in a position already equal to 1, thus game over)
     2. Treat Obstacles as Snakes, except that the game doesn’t end if you run into them (the snake just turns)
        1. Requires a Boolean isSnakeOrObstacle flag check
  3. **Input Handling:**
     1. Either use Gyroscope or RotationVectorSensor (since only one-dimensional rotation, there’s no need to even look at the other axes).