SD1 Assignment 5: Incursion Playable

Version 2 – updated Oct 17, 2019 – important changes are highlighted in green. Please un-highlight them once you’ve seen what they all are.

# Requirements

* (4 points) create a new **Incursion** folder / solution / project (in SD/Incursion), branched within Perforce from your Protogame2D project
* (4 points) class **AudioSystem** in Engine/Audio – major engine subsystem (like InputSystem & RenderContext), owned by App
  + See the *Instructions for using Audio.docx* document for instructions, found inside AudioStarterKit.zip (to be posted in Canvas)
* (5 points) class **SpriteSheet** and class **SpriteDefinition** in Engine/Renderer – provides uvTexCoords for any sprite index from a texture atlas
  + See SpriteSheetStarterKit.zip (toone be posted in Canvas) for details and expected usage
* (5 points) class **TileDefinition** in Game – instanced once for each TYPE of tile (grass, stone, mud, water, lava, etc.)
  + (2) The TileDefinition class itself owns these TileDefinitions as a vector of tile definition objects, indexed by TileType enum value:

class TileDefinition

…

static std::vector<TileDefinition> **s\_definitions**; // indexed by TileType enum, e.g. **s\_definitions**[ TILE\_TYPE\_GRASS ];

* + (1) Each TileDefinition holds any invariant (unchanging) data for that type of tile, e.g. sprite sheet UVs, tint color, isSolid, etc.
  + (2) Each individual Tile instance is flyweight, and holds **only** the following: IntVec2 **m\_tileCoords**, and TileType **m\_type**.
* (4 points) All Tiles (for the current map) are drawn textured, **using a single sprite sheet, in a single call** to Renderer::DrawVertexArray()
* (5 points) Various Map helper methods, including at minimum: **IsPointInSolid**(), **IsTileSolid**(), **Raycast**() and **HasLineOfSight**()
* (22 points) Six entity classes; each is its own C++ subclass that derives directly from Entity:
  + (6) **PlayerTank**
    - Pushed by Walls, Pushes Entities, Pushed by Entities
    - Driven using the Xbox controller left joystick (identical to description on SD1-A4: Incursion Foundation)
    - Drawn as two separate textured quads; PlayerTankBase.png (oriented by the tank) and PlayerTankTop.png (oriented separately)
    - While the Xbox controller right joystick is used (corrected magnitude > 0), the gun turns toward the joystick angle at a fixed rate
    - While the Xbox controller right joystick is not being pressed, the gun maintains its orientation relative to the tank body
    - The PlayerTank (body and gun) is a single entity (i.e. the gun is not a separate Entity!)
    - The gun’s turning rate is faster than the tanks, and is expressed as an angular speed relative to the tank body’s orientation
    - Fires a bullet each time the Xbox controller (A) button is pressed; bullet appears at the tip (forward “nose”) of the PlayerTank’s gun and flies forward (based on the gun’s absolute orientation) at a fixed speed until it impacts a solid wall or entity (see below).
  + (5) **NpcTurret**
    - Pushed by Walls, Pushes Entities, Not pushed by Entities
    - Player is “visible” if (a) within max range (15) **and** also (b) has line-of-sight (Raycast from NpcTurret’s center to player’s center)
    - Turns toward player if visible; turns counter-clockwise if not
    - Tries to shoot (1.3 second cooldown) if player is within +/- 5o of forward direction
    - Drawn as two images; EnemyTurretTop.png (drawn oriented/rotated) is drawn on top of EnemyTurretBase.png (NOT rotated).
    - Draws a pure red (untextured) “laser” line from the nose of its gun to the impact point of a forward Raycast (out to max range)
  + (5) **NpcTank**
    - Pushed by Walls, Pushes Entities, Pushed by Entities
    - Player is “visible” if (a) within max range (10) **and** also (b) has line-of-sight (Raycast from NpcTank’s center to player’s center)
    - If player is visible: turns toward player, drives forward if within +/- 45o; shoots (1.7 second cooldown) if within +/- 5o
    - else: turn toward random “goal orientation” while driving forward every frame; re-randomize goal orientation every 2 seconds.
  + (2) **Boulder**
    - Pushed by Walls, Pushes Entities, Pushed by Entities
    - No Update logic of its own
    - Deflects (reflects) bullets, but does not block line of sight / raycasts / NpcTurret laser beams
  + (4) **Bullet**
    - Not pushed by Walls, Cannot push Entities, Not pushed by Entities
    - Has a constant velocity once fired (i.e. flies in a straight line at a fixed speed)
    - Checks each frame if it overlaps a wall (and dies if so)
    - Checks each frame if it overlaps an Entity of a different Faction; if so, it damages that Entity and then dies
* (4 points) **Factions**
  + (1) Create an EntityFaction enum (in Entity.hpp) with three enumeration values: FACTION\_GOOD, FACTION\_EVIL, FACTION\_NEUTRAL
  + (1) Each Entity has an EntityFaction member indicating which Faction (team) it is on
  + (2) Bullets (shot by Player, NpcTurret, and NpcTank) only collide with (and deal damage to) Entities of a Faction different than their own
* (13 points) **Entity Management**
  + (3) Create an EntityType enum (in Entity.hpp) with one enumeration value per Entity type in the game; initially this will include: ENTITY\_TYPE\_PLAYER, ENTITY\_TYPE\_NPC\_TURRET, …\_NPC\_TANK, …\_BOULDER, …\_GOOD\_BULLET, …\_EVIL\_BULLET, and finally NUM\_ENTITY\_TYPES
  + (2) Create an “EntityList” or similar typedef (type nickname / alias) in Entity.hpp, as: **typedef** std::vector<Entity\*> **EntityList**;
  + (3) Each Map **no longer uses a single m\_entities list**; instead, each Map owns an array of EntityLists, one per Entity type:

EntityList **m\_entityListsByType**[ NUM\_ENTITY\_TYPES ];

* + (5) A Map::**SpawnNewEntity**() method (“factory function”), which can spawn a new entity of **any** type, as such:

Entity\* Map::**SpawnNewEntity**( EntityType type, Faction faction, const Vec2& spawnPosition )

* (7 points) Simple disc-disc and disc-AABB2 **corrective physics** (for entity-vs-entity and entity-vs-tile, respectively)
  + Each update, each Entity-pair combination are pushed off of each other as appropriate, based on each Entity’s physics flags:
    - m\_isPushedByWalls if true, this Entity is pushed out of solid tiles
    - m\_isPushedByEntities if true, this Entity will be pushed by other Entities (if they have m\_doesPushEntities == true)
    - m\_doesPushEntities if true, pushes other Entities (if they have m\_isPushedByEntities == true)
  + **Then,** **after** all entity-vs-entity corrective physics is complete, each entity with the m\_isPushedByWalls flag is pushed out of solid tiles
  + Each Entity also has a fourth “physics” flag, though technically this is currently only used by Bullet logic (and not for corrective pushing):
    - m\_isHitByBullets if true, Bullet entities (of other factions only!) can hit and do damage do this entity
* (3 points) Player death and respawn; NPCs lose interest in dead players
* (3 points) Multiple (3) maps; only entities on current map are updated & rendered
* (3 points) Each Map has an m\_exitPosition, located at the center of the (single) “exit” tile near the Map’s top-right
* (3 points) Change to next map (transplant PlayerTank entity from Map to Map) if m\_exitPosition is ever inside Player’s physics radius
* (3 points) Simple HUD shows Player extra lives (initially 3) as tank icons in top-left of screen space
* (3 points) Game over (lose) if player dies and no extra lives remain; Game over (win) if final map’s Exit is reached
* (3 points) Tanks (Player and NPC) overlapping one or more “mud” tile(s) have forward speed reduced by 50%
* (3 points) Appropriate sounds played for: player shoot, NPC shoot, player hit, NPC hit, player died, NPC died
* (3 points) All required textures and sounds are pre-loaded on game startup (and are never loaded during play)

# Submission

Submit your assignment by following the instructions above and checking in all the required files to Perforce (including a Release-built Incursion\_x64.exe), with the check-in comment “SD1-A5: COMPLETE” for the changelist you want me to grade. My Perforce changelist # **165377**

Also, in Canvas, you should submit a .zip file as follows:

* Submit a single .zip file to Canvas under the assignment.
* Your .zip should be named: **C29\_SD1\_A5\_LastnameFirstname.zip**

*For example, Jane Smith would submit a file named* ***C29\_SD1\_A5\_SmithJane.zip***

* Your assignment submission .zip file should contain the following:
  1. A **video recording** of you playing your game (and showing your code, as needed); be sure to visually demonstrate and verbally narrate each feature you want credit for
     + The video should be: 1920x1080 **.mp4** at 60 FPS, under 5 minutes and < 100 MB
     + Recommend you use OBS Studio (64bit) to record; make sure you check (watch) the video!
     + See the accompanying Demo video for an example of what’s expected here
  2. A **copy of this Word document**, with the following modifications:
     + Your submitted Perforce changelist # entered at the top of this section, at “My Perforce changelist # **???**” – this is the Perforce changelist # I should Get, run, test, and grade
     + Each line in the “Requirements” section with (X points) **must** be highlighted:
       - Fully completed requirements are highlighted cyan
         * for features you believe you’ve met/reproduced nearly exactly
       - Partially completed requirements are highlighted yellow
         * for features done but lacking or differing significantly vs. demo
       - Missing requirements are highlighted red
         * for features not implemented (not working or not attempted)