“Faith” engine specification

Implementations:

* Stage 1 – The Setting:
  + Basic scene setup with Ogre.
  + Basic physic world implementation for gravity with PhysX (NxOgre).
  + Mesh entry system, according to game specification.
  + Lights entry system.
  + Pre-input system (keyboard and mouse).
  + Camera and viewpoint entry.
  + Animation engine including animation blending.
* Stage 2 – The Game Engine part 1:
  + Creating position system on scene (255, 255, 5).
  + Creating object entry on fixed scene:
    - Every object has 3d position (x, y, z).
    - Object can (not must) have mesh.
    - Object must have deep (z) size (max 5).
    - Object has to determine if player will collision with it or not. If not the PhysX will ignore the object.
    - Object can have animations. Looped or run by trigger.
    - Object CAN NOT move. Moveable object will be called “Items” and won’t have fixed position like objects.
  + Displaying bitmap on plane in BG.
  + Creating player entry:
    - Player move with not fixed position (not on grid).
    - Tracking the player on fixed grid (x, y, z), so we know if he for example stand on trigger.
    - Player equipment (weapons, armor, etc.).
    - Player data (HP, Mana, Stamina, etc.).
    - Status (poisoned, slow falling, at jump(!) ,etc.).
    - Player mash entry.
    - Player animation.
    - Movement data (how fast can he run, how long and far jump, etc.).
    - Can move in deep only on 2, 3 and 4 layer. 1 and 5 are only for objects.
* Stage 3 – The Game Engine part 2:
  + Creating Triggers system, based on trippers system:
    - Position (x, y ,z).
    - What (player, NPC, item).
    - Condition (HP = 50%, Dead, Poisoned, etc.).
    - On (player, NPC, object).
    - Action (play animation, kill, damage, heal).
    - Active/Inactive.
  + Creating NPC:
    - Dialogs(?)
    - Some actions(?)
    - Animations.
  + Creating Enemies:
    - Based on NPCs.
    - Stats (HP, Mana).
    - (data based on battle system).
* Stage 4 – The Game Engine part 3:
  + Battle system:
    - (?)
  + Traps system:
    - Special reactivated triggers.
    - Die animation on PhysX’s collisions with player.
* Stage 5 – The Engine finalizing:
  + Maps load from file (headers design in future).
  + Implementing some kind of open source GUI (QuickGUI, MyGUI, CEGUI).
  + Building GUI:
    - Designing Manu.
    - Designing HUD (if any).
    - Designing Skills trees
    - Etc.
  + Designing keyboard controls.
* Stage 6 – Building The Game:
  + Create map designing software along with basic object, traps, enemies.
  + Designing map file format (headers) and implementing it to the engine.
  + Build levels.
  + Test for bugs.
  + Release.