# A2 Plan

## A2 Feature Set

(Black entries are game code; red entries are engine code)

* HUD shows base status bar and gun-with-hand sprite in screen space (depth test disabled)
* One Map is generated for each \*.xml file in the Data/Maps folder during Startup (Game::Game)
* Tiles and faces
  + Non-solid tiles have floor and ceiling (but no sides) quads, facing INWARD
  + Solid tiles have (up to 4) side quads, in each cardinal direction (North, South, East, West)
  + Hidden Surface Removal: solid tiles do not add NSEW faces if that neighboring tile is solid
* The world owns multiple named maps, and also knows which map is the “current” map
* Console command “map” which opens the named map (makes it the “current” map)
  + Provides hint with correct usage AND lists all valid map names
* DevConsole has friendly printouts for major parsing events (e.g. map loading, etc.)
* TileMaps reference “region types”, defined in MapRegionTypes.xml
* Current map (only) is drawn, in a single draw call
  + All visible tile faces are appended to a single mesh, which is then drawn
  + All world faces must be drawn using the same single texture (spritesheet)
* Various “Game Materials” are defined in MapMaterialTypes.xml
* All XML data is validated, and reasonable and helpful error messages are provided on parsing errors
  + On data error, print a red error message to console and force the console open
  + If possible, fall back to reasonable / error “data” (e.g. InvalidMapMaterial), and continue running
* Teleporter sound is played anytime the player enters a map (or is teleported)
* Audio files are loaded at program start; any expected audio files that are missing are a console error
* Each map specifies the player’s start position and yaw (pitch and roll are set to 0)
* Out-of-bounds (non-existent) tiles always query as being Solid (to help physics & HSR)
* FPS indicator should be displayed onscreen at all times
* F1 toggles debug draw of 2D debug overlay and 3D debug elements (world axes, compass)
* Most data-driven objects (MapRegionTypes, MapMaterialTypes, Maps, etc.) are constructed from XML
  + ...which means they are constructed at game startup, when we parse all XML files
  + Data is validated “as we go” while parsing the file / constructing the object
* The game loads a GameConfig.txt or GameConfig.xml file, which offers basic adjustable settings
  + GameConfig must be able to specify the starting map name

A2 Task Plan (listed in order of execution)

* Add a Console->Error function; like Console->Print but in red, and automatically forces console open
* Create a new Doomenstein project, cloned from Protogame3D
* Create & add to project near-empty .CPP / .HPP files we know we need, submit to source control
* Copy assets (images, XML, etc.) from A2 Demo and add to project & source control
* F1 behavior
  + Disable “F1 plays sound” code
  + F1 toggles g\_debugDraw
  + World axes & compass & UI debug overlay text are not drawn when g\_debugDraw == false
* FPS indicator is displayed onscreen (always)
* Load sounds (teleporter.wav) and textures (Terrain\_8x8.png, hud, gun, ???) at startup, confirm loaded
* World and Map
  + Simple World, Map (pure virtual), and TileMap (derives from Map) classes
  + World has only one map instance (so far): Map\* m\_currentMap = nullptr;
  + Game::Render calls World::Render calls m\_currentMap->Render;
  + TileMap::Render draws the test cubes from A1 (drawn however)
  + TileMap owns a single m\_mesh (of Vertex\_PCU)
    - Map has two pure-virtual functions: Render() const and UpdateMeshes()
  + TileMap::UpdateMeshes() **clears m\_mesh** and pushes a single fake “floor” quad (from 0,0,0 to 1,1,0) into m\_mesh, textured
  + TileMap::Render() draws m\_mesh in a single call
  + TileMap creates an array (or vector) of N MapTiles, by value, where N = MapWidth \* MapHeight
  + class MapTile: initially holds just IntVec2 m\_tileCoords and bool m\_tempIsSolid = false;
  + Map::Map constructor sets 90% of tiles to be non-solid, 10% to be solid OR outside tiles are solid, etc.
  + TileMap::UpdateMeshes() pushes 2 quads for non-solid tiles: floor and ceiling
    - Uses Test texture quads face INWARD (floor up, ceiling down)
  + TileMap::UpdateMeshes() pushes 4 quads for solid tiles; North, South, East, West
    - Uses Test texture quads face OUTWARD (east side faces east, etc.)
* TileMap::TileMap constructs from XML element of Map.xml document - load “TestRoom.xml” only
  + Parse only the root XML element and its “dimensions” attribute; make the map be that size
* .
* .
* .
* WEEK TWO.
* Engine utility function: Get all files in a folder matching a pattern “\*.xml”
* Multiple map support
* MapMaterialTypes are parsed and used
* MapRegionTypes are parsed and used
* Console “Map” command works correctly and robustly
* Good data validation and error behavior (Actually tested!) throughout
* HUD bar, HUD weapon, etc.
* ..
  + Asdf
  + Asdf
  + Asdf
  + Asdf
  + Asdf
  + sdf