*Are your player, items and enemies the same class, different classes in the same family, or completely different classes?*They will all be descended from a parent class. This is because they will all need to detect a collision, and have move and draw methods.

*What logic will you put into your Form class? What logic will you put into your Game Manager class?*The form class will contain the logic to create a GameManager, and at each timer tick tell the manager to drive the game logic.

*What class(es) do you need to implement the dungeon? Briefly explain the job of each class, list the data members it must hold, and the methods it must expose. How do the Dungeon and TileMap communicate?*

|  |
| --- |
| **TileMap** |
| TileList^ tileList; array<int, 2>^ map; Graphics^ canvas; |
| TileMap(TileList^ startTileList, Graphics^ startCanvas); void SetMapEntry(int col, int row, int tileIndex); Bitmap^ GetMapEntry(int col, int row); void DrawMap(); void GenerateDungeonMap(); bool isTileWalkable(int col, int row); |

The TileMap generates a procedurally generated dungeon and sets the tiles.

|  |
| --- |
| **TileList** |
| array<Tile^>^ tileArray; |
| TileList(); Bitmap^ GetTileBitmap(int tileIndex); void SetTileArrayEntry(int tileIndex, Tile^ tileToEnter); bool isTileWalkable(int tileIndex); |

The TileList holds a structure of Tiles, and can check if a Tile is walkable.

|  |
| --- |
| **Tile** |
| property Bitmap^ TileBitmap; property bool IsWalkable; |
| Tile(Bitmap^ startTileBitmap, bool startIsWalkable); |

A Tile contains an image and a data member indicating if the tile can be walked on.

|  |
| --- |
| **Sprite** |
| Graphics^ canvas; array<Bitmap^>^ spriteSheets; array<Point>^ velocityDirections; int nFrames; int currentFrame; int xPos; int yPos; int frameWidth; int frameHeight;  int SpriteDirection; int XVel; int YVel;  property Rectangle BoundingArea; property int BoundsAction; property bool IsAlive; |
| Sprite(Graphics^ startCanvas, array<String^>^ startFileNames, int startNFrames, Rectangle startBoundingArea); void draw(); void move(); void erase(Color eraseColour); void updateFrame(); void die(); void bounce(); void stop(); void wrap(); bool IsLegalMove(TileMap^ tileMap); |

**Hero**

**Enemy**

**Item**

*What data structure(s) do you need to hold collections of enemies and items?*The enemies and items will be held in their own array.

*Does the dungeon need pointers to its sprites? Why or why not?*

*Does the sprite class need a pointer to its dungeon? Why or why not?*

*What enum types (if any) do you need?*An enum for the sprite direction will be needed

*Does the player sprite need access to the collection(s) of enemy sprites?*No, however it will have access to a single enemy for detecting a collision.

*What class is responsible for creating the collections of enemies and items?*The GameManager will be responsible.

*If you are using an FSM, what class calls the FSM methods of the sprites?*The GameManager will call the FSM methods

*At each game cycle, you need to perform collision detection between the player character and each enemy and item in the dungeon. What class or classes hold a method to compare the areas of two entities to check for collision? What is the function header of this method? What other classes are involved in the collision detection logic?*  
The Player will contain a method to detect a collision with an enemy or an item.  
bool CollidedWithMe(Sprite^ otherSprite);  
The enemy/item will be passed in where the collision will be detected.

*If you are implementing Line of Sight what algorithm will you use? What methods are needed, and which class holds each method?*