One of the first challenges we faced was getting started. Consequently, we met an additional time to discuss how we should proceed and to examine some questions that arose since the project proposal. One topic of discussion was how to handle the map. We decided, based on its uniqueness, to create it as its own class with its own draw and update methods. We also discussed the size of the map and concluded that 2400 x 1800 pixels are sufficient. However, this is arbitrary and subject to change. Based on the types of objects in the game, we decided on the best draw order. From least low to high priority this is:

Even before beginning on the project, it became clear that many of our original labor divisions overlapped. For example, the original plan was for Matt to implement the map, and David to implement the restricted movement. However, this became a simple process for Matt to complete. We will attempt to maintain a division between tasks, but as more of this becomes apparent, it is easier to complete tasks on an “as needed and what is next” basis. Below is a list of the original milestones. Checkmarks represent completed tasks, while an ``X’’ is incomplete.

proceeds

We quickly realized that the game engine did not need the original planned base class, GameObject. This class, from which all other classes derived from, had one attribute for position and two methods for the size. Since our game is 2D, and everything is composed of sprites, we decided to change the base class to an abstract Sprite class. This has the same capabilities as the GameObject, but extends to cover textures, and covers most other classes from static sprites to animated characters. More important, we incorporation two new virtual methods that allow polymorphism for update and draw. Each sprite in the game now updates and draws itself. This removes large amounts of code from the base update and adds a new layer of abstraction.

Our changes from the GameObject base class to the Sprite base class lead to the development of the SpriteManager class. This is the first XNA game component of the project and it has several important functions in the game engine. First, it overrides both the update and draw method to call each sprites respective methods. Second, it ensures that the game draws each sprite in the correct order previously discussed. Finally, it adds that additional layer of abstraction and removes more code from the base update and draw methods.

We covered details of how and where the game engine will draw the map

We covered how we should handle the map drawing routine and the controls.

Draw order…

Size of map

Sprite controls

Base base class.