=+++++++ applesgame: Code Layout ++++++=

I realized I've pumped some code out without much explanation as to how it's all glued together. In here I'll give you a high level overview of how the pieces interact with each other so you can add changes easily.

As for any XNA Game, we have the Game1.cs class that is a Game Component. A game component in XNA is a class that is called in the game loop. Recall that the game loop consists of constant calls of system polling and update of our game logic inside the Update function of our class. As of right now, the Game1 class contains the logic to handle the transitions between Start Game, In-game, End-Game (not yet implemented). It's also drawing the initial screen falling apple, as well as the score. This is the place where you can add changes that have to be done in the game loop but are not related to sprites.

There is one other Game Component in charge of most of our game logic and that is SpriteManager.cs. SpriteManage is a class to manage our sprites and their logic, so it will initialize our sprites, do their update logic and draw them. Since our game consists of sprites and their interactions, this class will probably be the most important. As of right now, there is a player sprite and apples sprites being created inside their class. Their update methods are being called in our game loop to update these sprites (make them move) and basic collision detection is being done to detect when the sprites have reached ground level (so they no longer fall) or when the apples have collided with the player. If you need to do anything with sprites that does not require changing the sprites themselves, this is the place for it.

Sprite.cs handles what the sprites can do and is an abstract class so that specific types of sprites can implement the behavior. You will probably not have to touch this class but you may want to create subclasses of it if you want to create some sprites in the game. In order to make a specific sprite, you'll probably want to inherit from Sprite.cs and add any attributes or logic that your sprite needs. Recall that to add your sprite(s) to the game you'll be using SpriteManager. As of the of this writing, the Sprite class contains a lot of code that can be cleaned up, it certainly has more local variables than what was suggested in class. I think we should try getting some finished product working before cleaning it up though.

Inhering from Sprite.cs are UserControlledSprite.cs, AutomatedSprite.cs, UserAnimatedSprite.cs, and AppleSprite.cs. UserControlledSprite is being used for to represent the player and AppleSprite is being used to represent falling apples. The other two are being unused and were actually made for experimentation (UserAnimatedSprite is not even finished). UserControlledSprite handles user input and allows players to control the sprite. UserAnimatedSprite was supposed to extend on this functionality by playing different animations depending on different user inputs, but I think it'll be easier to make appropriate change to UserControlledSprite instead. AutomatedSprite is similar to AppleSprite in that it's supposed to do automated logic, this one along with UserAnimatedSprite will probably be removed, so don't worry about them.

A few other things to note. Notice that there are two project solutions inside our solution explorer. This is because our content is handled in the Apple01Content solution. You can add folders and files to this project for any game assets such as graphics, sounds, etc.

Anyways, that's all for now.