Lucas Connors

ISU#1 Write-Up – Game Design Part I

In order to properly design a computer game or a video game, the developer(s) must go through the three crucial stages in creating the basic structure of a game. The first step of these is called Pre-Production. This is the area where the developer(s) (hereinafter being referred to simply as developer) organized all aspects of the game and how they will work. The second step is Production. This step is where the basic mechanics of the game are all put together in something called an engine. The final step of development is the Post-Production stage. In this stage, the game comes all together as the specific plans in the Pre-Production stage are put into the game. Glitches and bugs are also corrected, and final details are added.

Pre-Production

This write-up will focus primarily on Pre-Production, and the earlier procedures of Production involved in developing a common maze game. To begin with Pre-Production in developing a maze game, first of all you must establish what type of maze game it is. Although this decision can certainly be interpreted several ways, there are three primary concepts that I recommend should be determined as the first step in creating your game.

Establishing the primary concepts of your game

First, it must be determined what general concept of a maze game you are planning on creating. For example, you may create a maze game that follows a specific path from the beginning to the end. Your maze game may be an open-world type simply where the player has to choose which rooms to walk into next. You may have a combination of both, or something entirely different. For the purposes of following a static example, I will carry through this write-up using examples from a maze game where the player may choose multiple paths throughout each level, but as a result will reach the same destination at the end whether or not they received the greatest reward as a result of the outcome. Traditionally, this is referred to as a "Limited Sandbox Maze Game". Finally, once this aspect of the game has been established, we can then determine the theme overall. Maze games can have an unlimited amount of different themes to them. Generally, the theme of the game will reflect a combination of your audience, the resources you have access to (or that your Art Department can develop, if you are so lucky as to have one at your disposal). You may decide to create a science-fiction themed maze game, or possibly a mythical-fantasy themed maze game. For the purposes of my example, I have decided to create a medieval-themed maze game primarily due to the visual resources I have available. The final main concept that I recommend deciding early on is how the Goal and Reward System will work. At this point, I would like to take a few moments to explain what a Goal and Reward System actually is. One of the most important concepts in creating a game is determining how the player will be directed to completing his or her goals, and then how the player will be rewarded. Your game may not set out a specific goal after each goal is completed, but not necessarily lay out what goal is to be completed next and leave the player to find that out (this is usually only an appropriate choice if the goals are relatively obvious). In my example, I will choose to implement a system where a set of a few goals can be completed and the player will be rewarded differently occasionally depending on the choice they made. However, in the end, they will still reach the same final destination at the end of each level. Traditionally this is called a "Limited Tree System". In the example, the goals and rewards are represented by the rooms they enter, but of course this can be done in limitless ways. Often the goal and reward system is heavily affected by the general concept. On the following page are some examples of the most common Goal and Reward Concepts.

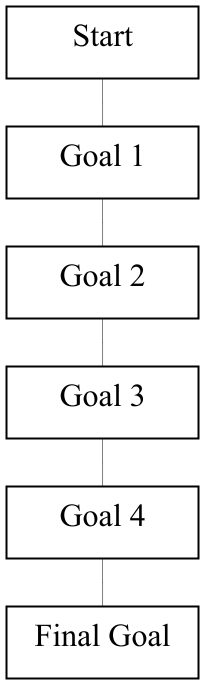
Writing a story

Now that we finally have the general concept established, the next step in making a maze game is to write the story. At this stage, you will probably feel obligated to create the characters as well. This is probably a good idea. Your story really only needs to consist of the back-story which explains where the main character is and why he or she is there, as well as the motive for the player to complete the goals set out for them to reach the end of the game. It may also be a good idea to explain what happens at the end of the game, but if all the rest of the story is interactive, it may only be necessary for you to establish the prologue of the story (called the premise). Here is an example of a story for a maze game fitting the description established so far:

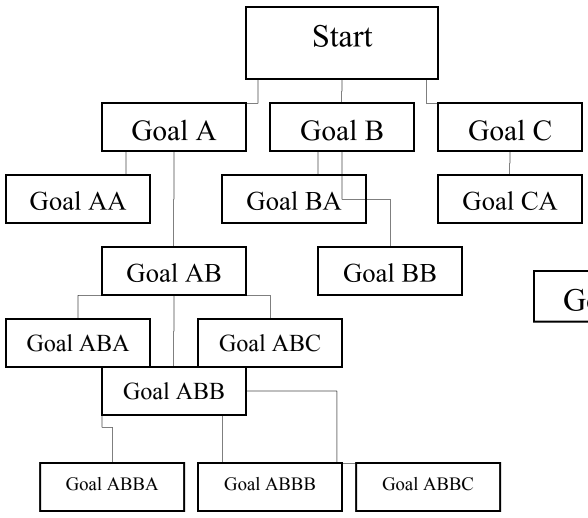
"*You are a knight by the name of Ethan Goddard. King Sullivan has praised you as his greatest knight, but the King has decided to invade your homeland. Because you disagree with this, you decide to defend your homeland and end up defeating all of the King's "good" men. This infuriates King Sullivan and he decides to throw you in the labyrinth. Your only chance for survival is to reach the end of the labyrinth and defeat the King.*"

Drawing sprites and backgrounds

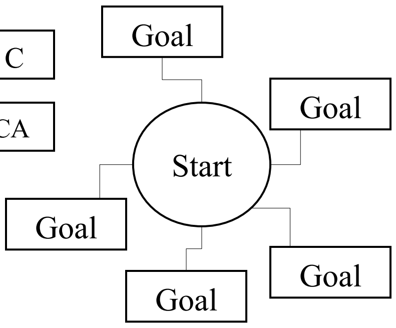
The next step of preproduction in your game is actually drawing the sprites and backgrounds that will be used in your game. A sprite is an image of a character or object in a frame of animation. This could be an image of a wall object, or it could be a character walking at its forth frame of the walking animation. Backgrounds are relatively self-explanatory. They are images that are behind everything else in the room. They provide a good indication of the setting. You can draw these in computer software such as *Adobe Photoshop* or you can physically draw them on paper, and scan them in and edit them, but in the long run they need to end up in electronic form. Because you are going to want a transparent background around your sprites, I recommend that you just make the sprites in electronic form in the first place, even if you need to use a tablet to do so. Backgrounds tend to be less strict on this type of detail so scanning them is less of an issue. For the purposes of my example, I have used royalty-free sprites to which I have permission to use and have given credit in the game for the use of their resources. This is always of course an appropriate choice considering the conditions that you receive permission and give credit for the other person's work.



Linear System



Tree System

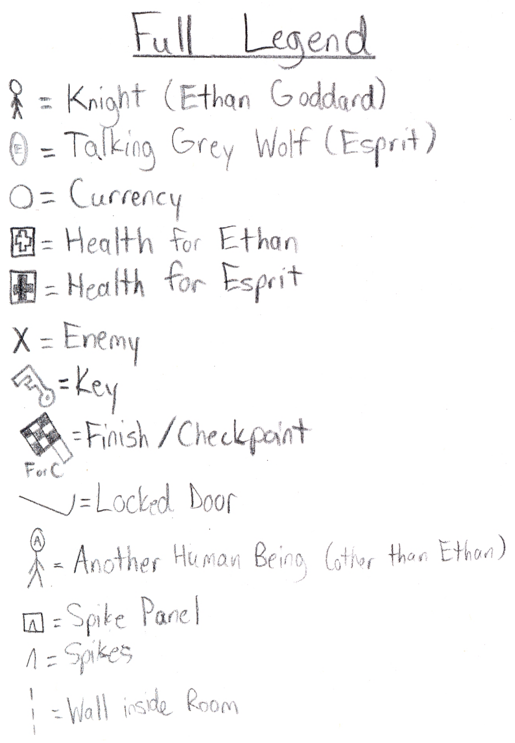


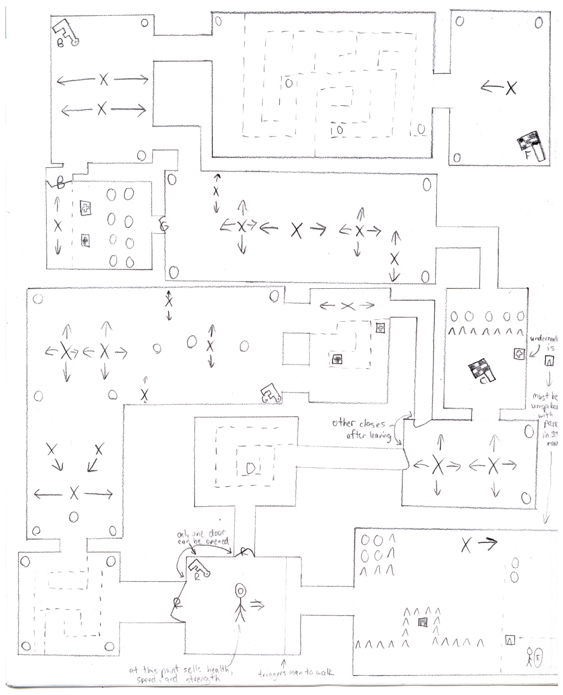
Wheel System

You can always finish less important sprites and backgrounds later, but once you have enough to actually create your game, you can get started. Keep in mind that games tend to be made by several departments all at the same time, so there really isn't any correct order to creating a game except for the very first step, and even then they often tie into the second step immediately. As a matter of fact, several games are created with a completed prototype without even deciding any details on the characters and story at all. They would just use blocks of different colours and sizes to distinguish different objects, but for the sake of making this write-up easier to understand, we are going to visually see the characters and objects earlier in the production of the game. As well, I am going to recommend that you compose, or otherwise gather royalty-free music for your game at this point, although certainly this could be done at any point, even at the end of the development of the game.

Designing level maps

Now if you weren't already having fun at this point, get ready for some serious fun! This step of the Pre-Production stage for making a maze game is the most entertaining for many people. At this point, we are going to draw maps on paper to decide what each level will look like, and where everything throughout the level is placed so that making the rooms later will be a much easier process. I recommend creating a legend for yourself to distinguish important objects from each other. Below is an example of one of the levels I have designed for my game as well as the legend I used for the objects in all of the rooms.




other closes after leaving
only one door can be opened
at this point sells health, speed, and strength
triggers man to walk
underneath is
must be unspiked with panel in

Production

Now that we have all of the necessary planning completed, we can start to actually develop our game in *Game Maker*. The first step you need to do in production is to open the software, *Game Maker* and enable Advanced Mode.

Starting the project

Once *Game Maker* is open, we are going to make sure that you are currently in Advanced Mode in *Game Maker* as that will allow us to access some more advanced features in the program which you will need. Select "File" in the menu bar at the top of the *Game Maker* window and check that the option "Advanced

Mode" has a checkmark beside it. If it does not, select that option (see Diagram 1).

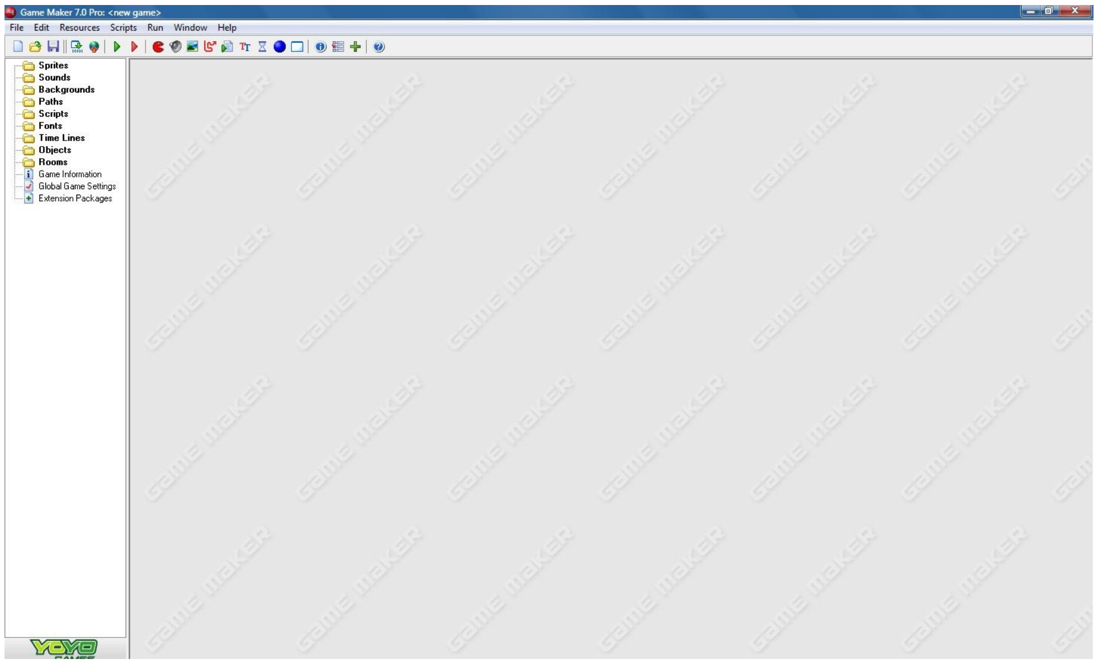


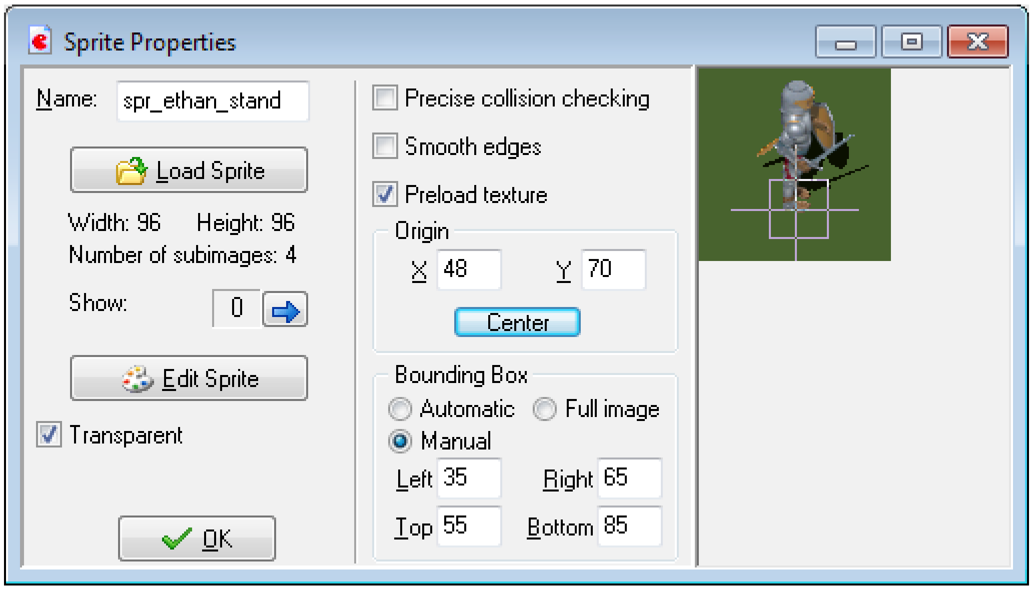
Diagram 1: This is what your window should look like if Advanced Mode is enabled.

Now, save a blank document where you can find it later. It is easier to save a blank document now so you can just click the "Save" button later when you're in the middle of your work without being interrupted to declare a path and filename. As a reminder, always save your work constantly! You never know when the power may go out!

Importing sprites

The next step to creating your game is to import the sprites you drew in the Pre-Production stages. First we will sort them by groups; this is extremely useful if you have many sprites. You may decide to sort your sprites by character if you have several sprites for each character. To create a group for sorting your sprites, right-click the sprite folder in the resources menu on the left and select "Create Group". Now you can name the group whatever you want. When you are ready to add a sprite to one of your groups, right-click the desired group and select "Create Sprite".

A window will then appear where you can select all of the settings which you may want for your sprite. First click the button, "Load Sprite" and navigate to the sprite which you wish to put in. Select that sprite and click, "OK". At this point you should give you sprite a name. **Do not use any spaces or symbols. Use underscores (_) instead of spaces.** Also keep the name "sprite" at the beginning of the filename so that *Game Maker* does not confuse your sprites with your objects. A name such as "sprite_ethan_stand_east" may be appropriate (see Diagram 2). Now you may change the various settings regarding the sprite including the origin, transparency and other options. You can also edit the sprite in the Game Maker Sprite Editor which is similar to that of *Microsoft Paint*. For each of your sprites, I recommend that you select the "Center" button so that the origin changes to the center of the sprite. This makes collision-detection easier to work with later. If you wish to have the background of your sprite be transparent, consider that *Game Maker* uses the bottom-left pixel of your sprite as the background colour.


Sprite Properties

Name: spr_ethan_stand

Width: 96 Height: 96
Number of subimages: 4
Show: 0

[✓] Transparent

[ ] Precise collision checking
[ ] Smooth edges
[✓] Preload texture

Origin
X 48 Y 70

Bounding Box
Automatic Full image [✓ Manual]
Left 35 Right 65
Top 55 Bottom 85

Diagram 2: This is the Sprite Properties window.

You may notice that in my sprite, I have set a manual bounding box around the character's feet with the origin in the center of that bounding box. The reason why I have done this is due to the player's perspective on the dimensions of the game. As you can see by the sprite I have, the game will appear to be 3D, but I am actually programming all of the objects in 2D. If I do not specify the bounding box this way, the game will recognize a collision with an enemy and the main character when the enemy is coming from above and their feet just skim the top of the main character's head. If you imagine this for a moment, you will recognize that this does not make sense. Hopefully, you will not have to worry about this issue too much because as a starting maze game, I certainly recommend using simple bird's-eye-view low-quality sprites to avoid problems such as these.

Creating a test room

Once you have imported all of the sprites for your character, (at this point you should have at least sprites for standing and walking in all 4 co-ordinal directions) you can begin the process of creating a test room.

Create a test room by right-clicking the Rooms folder in the resource tree and selecting "Create Room". Select the "Settings" tab in the Room Properties window and change the name of the room to something appropriate such as "room_test". I also recommend changing the caption of every room to the name of the game that you're creating.

Now, before you do anything more with the room, you really need to create objects to go *into* the room.

Create an object by right-clicking the Objects folder in the resources tree and selecting "Create Object". Start by creating an object for your main character. Give your object an appropriate name, such as "object_ethan". Set the default sprite to your main character's standing to the right sprite, such as "sprite_ethan_stand_east".

Select the "Add Event" button. Select the "Create" button to add a Create event.

Select the control tab to the right of the Object Properties window. Drag the "Execute Code" action icon (the icon that looks like the one to the left) into the Actions column of the Object Properties window (see Diagram 3). This will add an Execute Code action to the Create event.

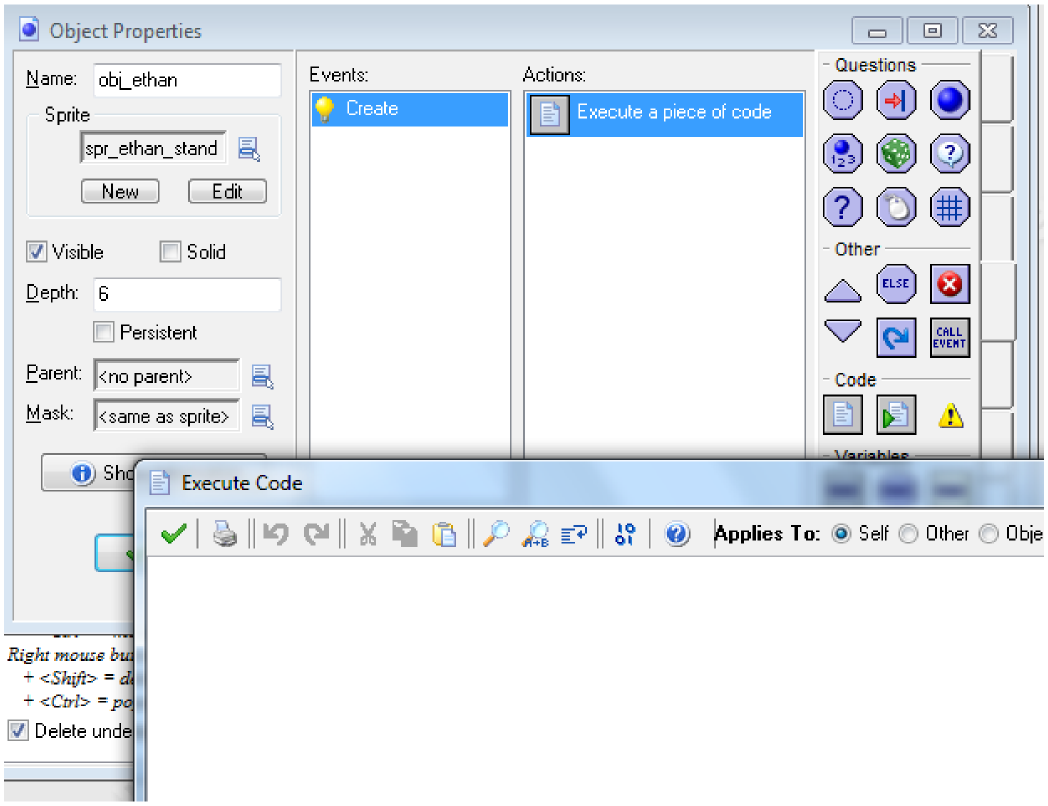


Diagram 3: This is the Object Properties window.

A code window will automatically appear. When this happens, type the following code to initiate your character:

```
direction=270;
move=1;
```

Close the Code Window by clicking on the green checkmark. Note that you can open the code back anytime by double-clicking on the action in the Actions menu. The code above sets the character to face in the downwards direction. In the next bit of code we will add, we will set the sprite of the character to face the correct direction depending on the way they are facing. This way, with the combination of the code we entered above, and the code we will enter next, the character will automatically change from facing to the right as we put the default sprite to, to facing down as our code instructs.

Select the "Add Event" button. Select the "Step" button to add a Step event.

Select the control tab. Drag the "Execute Code" action icon into the Actions column of the Object Properties window. Add the following code to the Code Properties window to make the character face the correct direction:

Note: The names of the sprites in the code below match the names of the sprites for the example I have created. You will need to change the names of **bolded** resources to the actual names of your sprites for your code to work.

```
//Set sprite

//Standing
if (speed=0 and move=1){
    if (direction=0){
        sprite_index=spr_ethan_stand_east;
    }
    if (direction=90){
        sprite_index=spr_ethan_stand_north;
    }
    if (direction=180){
        sprite_index=spr_ethan_stand_west;
    }
    if (direction=270){
        sprite_index=spr_ethan_stand_south;
    }
}
//Running
else {
    image_speed=1;
    if (direction=0){
        sprite_index=spr_ethan_run_east;
    }
    if (direction=90){
        sprite_index=spr_ethan_run_north;
    }
```

```
        if (direction=180){
            sprite_index=spr_ethan_run_west;
        }
        if (direction=270){
            sprite_index=spr_ethan_run_south;
        }
}

//Snap to grid
if (place_snapped(60,60)){
    speed=0;
    move_snap(60,60);
}
```

In the last bit of code above, I have set the grids to be 60x60, your grids will likely not be this size, and most likely be approximately 32x32 if you use professional sprites for simple computer games.

Now that our character will properly snap into a grid, we need to get our character moving! This next bit of code will do that.

Create a new event similarly to the way you created the previous events for the character object. This time however, add a Keyboard event. In the submenu, select the key you wish to use to move the character left. The left arrow key is always a safe choice for a simple maze game.

Once again, add an Execute Code action. Add the following code to allow the player to move left:

```
//Move left
if (move=1 and place_snapped(1,60)){
    direction=180;
    speed=5;
}
```

Similarly we are going to follow the last two steps for the up, right, and down directions.

The code for moving up is as follows:

```
//Move up
if (move=1 and place_snapped(60,1)){
    direction=90;
    speed=5;
}
```

The code for moving right is as follows:

```
//Move right
if (move=1 and place_snapped(1,60)){
    direction=0;
    speed=5;
}
```

The code for moving down is as follows:

```
//Move down
if (move=1 and place_snapped(60,1)){
    direction=270;
    speed=5;
}
```

Now that our character is programmed to move around, let's add the character to the room we made earlier.

If the test room is not still open, double-click on the room in the Resource Tree to open it. Select the "Objects" tab. If your main character is not selected by default, click in the large blank grey area in the "Objects" tab or on the menu icon to bring up a menu where you should have an option to select your main character. Select your main character object. Single-click anywhere in the room to add your main character to the room.

Now you can test the game by clicking the green arrow in the main toolbar, or by clicking F5 once the Room Properties window has been closed. If you have followed all of the steps correctly, your main character should now move around the room nicely with animations!

Congratulations! You have now created the basic engine for a maze game. At this point, you are free to add more to the engine such as allowing the character to attack, collect items, or you could even create a second character! In the write-up of my second Independent Study, I will guide you through the Post-Production and ending stages of a maze game. Stay tooned!