UW–MADISON COMPUTER SCIENCES

CS 202 Spring 2010: Project 1

Programming Project 1 : Due Thursday 2/24 (Midnight) - NO LATE PROJECTS ACCEPTED

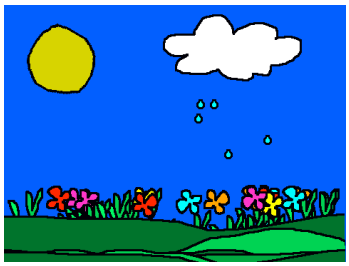
The goal of this project is to create an animated music video or animated poem.

Your animation can be anything you feel that illustrates the music or poem you've chosen. Your animation could be artistic, abstract patterns. Your animation could be specific pictures that act out the action. Your animation could be text and symbols that match the music or poem.

Of course, the images that you choose should match in some way with your music or poem; they should match either in topic or in mood. You might want to think about ways in which you can change the pace of your animation to match the tempo of the music. To personalize your project, we strongly encourage you to draw you own costumes or import images that are not part of the standard Scratch installation.

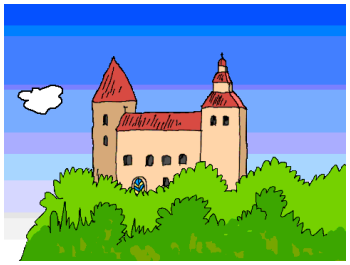
Your animation is **not** expected to be interactive; that is, the actions of the user (e.g., clicking the mouse or typing characters) are **not** expected to change how the animation behaves. Your animation can have a random component: the exact location or color or shapes of visual effects can be slightly different from run to run.

Probably the best way to convey what we expect from your project is to give you some examples. We assume you will investigate all of these samples before developing your own.

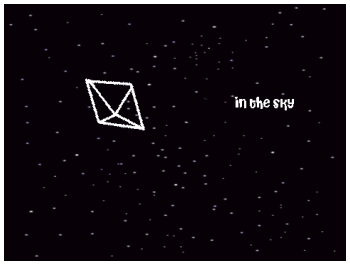
1. 
   Remember the video Outside to Play that I showed in the first Lecture? I think this is a great example of a project you can rather easily implement. You'll want to download the code from the Scratch website to see how they make it look like different flowers are growing.

2. 

On the Scratch website, Contrails shows two jet planes in the sky as a road and trees cycles underneath. It contains some neat effects. Make sure you download the code to understand how they make that scrolling happen!

3. 

This project plays its music in a different way. Instead of playing a recording, the authors are using the "play note" blocks within Scratch to compose music. The animation that goes along with it isn't bad either!

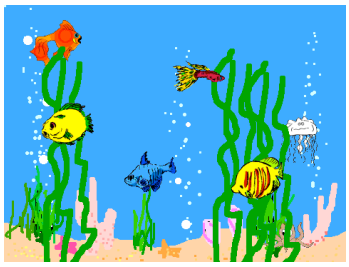
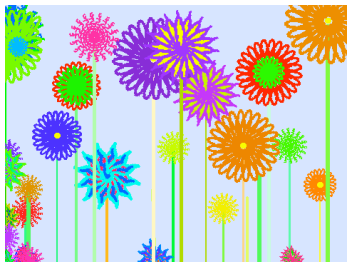
4. 

The Scratch application includes a number of interesting **Examples** that you can run when you click under **File** and **Open...** (click on the **Examples** box on the left-hand side of the pop-up window). Under the category **Music and Dance** you'll see a lot of great examples. I particularly like **4 TwinkleTwinkle**. You can imagine this is an example of an animated poem.

5. 

You should also look at a much more sophisticated interpretation of the music video. You can watch this Ted talk Jakob Trollback rethinks the music video in less than 4 minutes. This music video wasn't implemented in Scratch, but it will probably give you some great ideas.

At a minimum, you should absolutely investigate the four examples given above, but there are other interesting projects as well.

For example, we suggest exploring more of the **Example** projects included with Scratch. Under the **Animation** category, you might find inspiration from **6 Aquarium** (shown above). Under the **Interactive Art**, I like **7 Garden Secret** (shown above). These projects are a little simplistic, but you can still use ideas from them as a starting point.

The Scratch web site has many other interesting examples as well. We suggest looking for projects that have been tagged as **Music**

Finally, if you are interested in creating a visualization of a poem, this very short Ted talk called "Rives tells a story of mixed emoticons" will probably give you more ideas.

We expect that you will put many hours of thought and work into this project. Do not wait until the last minute to start implementing your project! Code written in a rush rarely works the way you hope it might! If you start your project promptly, it will be much easier to get help and advice from the Instructor and TA.

In terms of the final product, we expect that you will create a project of similar complexity to the first five examples given on this page.

Specification

The only requirement for this project is that you create a visual effect that complements either some accompanying music or a poem.

If you choose to accompany **music**, you can choose any music you like. You have a number of options here. First, you can import any music file you have access to (e.g., .mp3 or uncompressed .wav or .aif or .au file formats, but not .mp4) into Scratch. Second, you can use the Sound clips that are included with Scratch; however, none of these clips are very long, so you are likely to need to play several clips one after another to get a "song" worth animating. Third, you can use the "play note" blocks to compose and play your own music in Scratch (like the Castle example above). Fourth, if you are very brave, you can always record yourself singing!

If you choose to illustrate a **poem** you can choose any poem you like. You can write a poem yourself or you can use an existing one. You can record yourself reciting the poem and play that recording simultaneously with your visualization. Or, you can have the text of the poem appear on the Stage.

Implementation Hints

As always, you should strive to write code that is easy for others to understand and easy for you to modify. This usually corresponds to writing code in a compact manner and code that is identical across all Sprites that

are doing the same actions (with perhaps only local variables and initialization code differing across similar Sprites).

We recommend writing small amounts of code and immediately testing that code to see that it works correctly before writing more code. Get each step working correctly before you move on to the next step!

Documenting your Code

Part of your grade will be based on how well you document your code.

Code must be documented so that others can understand how it works. You will find documenting your own code useful: it is very easy to forget how code operates, even when you are the one who originally designed it! Because documentation is so important, your project grade will be partially based on the quality of your documentation.

Documenting the behavior of your code has three components:

1. **Use good naming**. All of your Sprites (and their Costumes) should have descriptive names. The name of a Sprite can be changed in the text box above the Tabs for Scripts, Costumes, and Sounds.

   All of your variables should have descriptive names. Variables which are accessed across multiple different scripts or Sprites should be named with the convention "First Letters Capitalized. Local variables which are not used outside of one script should be in all "lowercase".

   All of you messages (for broadcasts and receives) should have descriptive names as well. For a message X, to help show which Sprite sends it and which receives it, it can be useful to name the message "Sender : X : Receiver" (where Sender and Receiver are the names of those Sprites, respectively). As a short-cut, if a message is both sent and received within a single Sprite, you can omit "Sender" and "Receiver". If a message is sent by multiple Sprites (or received by multiple Sprites), you don't have to list all Sprites; instead you can just say "Many". For example, if Sprite1 sends the message "Game Over" to multiple Sprites, you should name that message "Sprite1: Game Over: Many".

2. **Write Comments**. Each Script that changes the value of Variables that are accessed in other scripts should describe its usage. The Comment should be connected to the Script. The Comment should describe every Input Variable and the assumptions that are made about those variables. The Comment should describe every Output Variable and how they will be set and what different values mean. (You don't need to make a comment for Scripts that don't access global variables, unless you want to!.)
3. **Project Notes**. Every Scratch project has Project Notes associated with it. These notes should describe how one can use this project (i.e., the basic rules for playing the game and how to interact with the program). The note should also describe any known bugs or problems. Project Notes can be written from the "File" pull-down menu. **Your project notes should give the name and author of the music or poem you have incorporated.**

Developing your Code

As always, programming assignments and projects in this class should be done on your own. You may ask other students in the class questions, but you **may not share code** with anyone in the class. You **may not use existing code** that you find elsewhere, including the Scratch website. You may look at the behavior of existing Scratch projects for inspiration, but you should develop all of your code as a completely new project and not modify, re-mix, or build from any one else's code.

The Instructor and the TA are very happy to give you suggestions on how to implement your ideas. We won't necessarily give the answer, but we will try to guide you to a reasonable implementation. If you have bugs in your code (i.e., it isn't behaving like you expect), we are happy to take a look and see if we can see the problem. But, again, don't wait until the last minute to do your project if you are hoping for any advice!

Grading your Code

Your project grade will be based on the following components:

- **Creativity/Effort (30 points)** Does it look like you put effort into the project? Do the backgrounds, characters, and objects have a logical and/or interesting theme?
- **Documentation (20 points)** Do you have good, descriptive names for sprites, costumes, variables, and broadcasts? Did you write comments where appropriate and use the project notes?
- **Specification (15 points)** How closely did you follow the specification for the project? Did you implement all of the features we asked you to?
- **Code Style (15 points)** Do you use the correct programming structures?
- **Demo (20 points)** Were you able to explain how your code works to the TA in your demo? Did you show your project to the class?

Turning in your Project Code

You should upload your project Scratch file (ending with the extension .sb) to your Learn@UW Project1 folder. If you have any problems uploading your code, send us email right away!