CSC 484
Arion Clarke
Ryan Murphy
Tawa White
3/19/2007
Final Project Writeup – Game Creator

Project Introduction

The purpose of our project is demonstrating a user interface for Game Creator. The Game Creator application is a rapid game development and learning tool for casual and mobile games. The goal of the project is to establish a user interface that allows game developers and students alike to quickly develop simple games with little or no exposure to the source code.

Motivation for Game Creator

The primary motivation for Game Creator comes from the discussion of a possible program at Cal Poly for game development as either a major in its own right or an elective focus. We believe that such a program would benefit from a tool like Game Creator, especially for freshmen students taking a 100 level game development sequence. Our hypothesis throughout the project has been that having a tool that allows students to create simple games without the need to write code directly emphasizes the process of creating games over writing lots of code. This, in turn, will make game development for incoming freshmen more fun and increase retention.

We think that Game Creator will uniquely fulfill this need by providing a method for developing simple games without the use of code. Instead of code, the main method for designing and developing games is through state diagrams.

Similar Products

Of course, there are a couple of other products out there that provide users with the ability to produce casual games without use of code the first is Game Maker [1] and another one is through DigiPen's Project Fun online game program. However, DigiPen's SDK is for registered users only and only provides their SDK to paying members of their online program. As such, we focused our product evaluation based on the comparison with Game Maker.

However, we feel that these products don't fully satisfy the goals and requirements of our project. While, they provide many of the same features that we envision in Game Creator, we believe that the usability of these products could be improved by providing a different mental model for game development (refer to our usability section for more information).

Game Creator Requirements

The main requirement of Game Creator is that it provides students with a means of developing simple games. We want to provide a product that makes it fun and simple to make games. Furthermore, another goal of the project is to emphasize the creative aspect of the game development process and de-emphasize that use of code. As such, we require that it is simple to make simple games without the need to write code.

For the purposes of the User Centered Design course however, we recognize that fulfilling this requirement completely by creating a fully implemented product is both lofty (considering the time constraints) and moves away from the course objective. As such, we have proposed the more manageable goal developing a user interface prototype and flushing out the conceptual models related to our project. Again, this is the driving force behind our project. We want to know how our conceptual model (state driven development) compares to other models for developing games (specifically, event driven and code driven development).

For the purposes of our project, we assume that there will be two types of user. The first type is first year students learning how to develop games. We expect that these students will have little or no programming experience. The second type is experienced game developers, who need a tool for rapid game prototypes. Note that we are more concerned usability of our product as far as the students goes.

Additionally, we intend our product supported on Windows XP initially.

Game Creator Overview

For the overview of our project, we refer to our storyboard. Here is a brief discription of our storyboard though.

Again, we chose to develop around storyboard around one use case. The task is to build a simple game for mobile devices, such as cell phones, palm platforms, and simulators.

Storyboard Outline

The storyboard presentation consists of the following steps for completing our task:

1) Establishing shot
2) Add game state
3) Completing game states
4) Go to actors
5) Add Actor
6) Actor with states
7) Actor with trigger
8) Completed actor list
9) Preview of actors
10) Running game

For more information and the actual storyboard power point slides, refer to the corresponding documentation

Game Creation Outline

From this, the task of creating a game with Game Creator consists of the following steps:

1) Create New Game specifying name and screen resolution
2) Add game states (*Main Menu, Game Play, Win*)
3) Add Background to each game state
4) Add menu items to *Main Menu* menu state
5) Add Actors (*Ship, Enemy, Bullet)* and corresponding actor state machines/diagrams
6) Place Actors on background for starting locations in the *Game Play* interactive state
7) Add text for *Win* dialog state
8) Run Game

Note that this is still a very high level outline of how to create a game but it covers the basics.

Data Gathering

The goal of our data gathering is to determine if students are more inclined to use state diagrams or code based techniques (i.e., pseudo code) to solve simple state based problems in the domain of casual games. This information is important due to the fact that our project is based on the hypothesis that incoming students will have an easier time developing simple games using state diagrams than coding directly in a given programming language (e.g., C++, Java).

Data gathering techniques

Our team chose to collect our data in the form of survey. The target audience of our survey is CSC 101 students.

The survey consists of the following:

1) An example problem with two solutions: state diagram and pseudo code. This is to ensure that the participants have a baseline for what we expect.

2) Another example problem in which the participant is asked to solve the problem using one of the two techniques (state diagram or pseudo code). Note that we are not evaluating the solutions for accuracy, but instead, we are concerned with which technique they choose.

3) Several questions asking the participants what their experience level is with state diagrams and coding respectively.

Initial results

We used the lab data gathering exercise in lab to both test out our survey before we pass use with CSC101 students. This information was helpful in flushing out the questions and provide baseline from students with more experience.

Some of the information that we found useful had to do with our problem description and definition. For instance, we found that several participants thought that they needed to solve it using both techniques, rather than choosing only one. Also we found that some of our naming conventions needed to be more descriptive. Finally, we realized that it is useful to find out if the participant naturally thought about the problem in one way over the other.

Results

We surveyed 44 CSC 101 students. We found that 57% of the students chose the state diagram over pseudo-code. This was not as high as we may have liked but it still shoes tendencies towards state diagrams.

We asked the students how easy they thought it would be to perform some simple functionality by creating a state diagram vs. using pseudo-code. Among the 44 students we found that the average result was in fact average or a score of 3. For the specific results, we had an average of 2.91 for state diagrams and 2.95 for pseudo code. As you can see there is a slight tendency towards state diagrams being easier to use, but this is hardly worth mentioning. These questions were on a scale of 1-5 (where 1 = Easy and 5 = Hard).

As expected, students were more comfortable with pseudo-code than state diagrams. We had an average of 1.34 for state diagrams and 2.70 for pseudo-code. These questions were on a scale of 1-5 (where 1 = None and 5 = Expert).

For the last question of the survey, we wanted to get an idea of what level of interest there was in this sort of class. Our average across the 44 surveys was 3.39. This means while some people were interested others were not. Another way of looking at the results to this question is the median value. And here we got a 4. These questions were on the following scale, 0-None, 3-Average, 5-I want it now.

Numerical Results

	Average	Median
Problem 1 (1=State 2=Pseudo)	1.43	1
Question 1	2.91	3
Question 2	2.95	3
Question 3	1.34	1
Question 4	2.70	3
Question 5	3.39	4

For more information, please refer to our writeup on the Data Gathering assignment (this is attached).

Usability Evaluation

The overall objective of our usability evaluation is to determine if our project (Game Creator) is usable as a tool to develop a simple game. That is, we want to know if the Game Creator can be used for creating a simple game by students in a 100 level game development course. As such, our evaluation is focused on the task of going through stages developing a game.

Furthermore, throughout the paper, we base our analysis around specific conceptual models for developing a game; specifically, comparing event based design and development to state based design and development.

However, since we don't have a fully implemented product yet, we are referring to the Game Maker product in place of our own Game Creator.

With this in mind, we decided to take two approaches to our usability evaluation. The first is a heuristic evaluation of Game Maker preformed by each member of our team. The second consists of an expert evaluation of our product based on a walkthrough of our product with some comparison to Game Maker where appropriate. Each evaluation method is described as follows:

Heuristic Evaluation

We based the heuristic evaluation on the ten principles for a user interface design outlined on http://www.useit.com/papers/heuristic/heuristic_list.html. We used these heuristics to analyze the usability of Game Maker for doing the task of developing a simple game. Here are the ten principles that we worked from. Note that not all of these evaluation heuristics map directly to our own task, but instead, we used this list as a set of heuristic guidelines. Also note that our team members have different experience levels with Game Maker.

These ten heuristics were taken directly from the above website:

Visibility of system status

The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

Match between system and the real world

The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

User control and freedom

Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

Consistency and standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

Error prevention

Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

Recognition rather than recall

Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

Flexibility and efficiency of use

Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

Aesthetic and minimalist design

Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

Help users recognize, diagnose, and recover from errors

Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

Help and documentation

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be

easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Results

Evaluation done by Arion (1 game made):

1) Visibility of system status
    a. There does not seem to be any major delay in Game Maker as you try and work.
2) Match between system and the real world
    a. Some of the terms have been used in industry for many years (eg: sprite) while others seem to have been added by them to encapsulate functionality (eg: room). In Game Creator we have strived to use more intuitive wording while staying true to industry.
3) User control and freedom
    a. While most GUI applications support undo and redo functionality, Game Maker does not. I am also not sure if we had planned to implement an undo and redo feature in Game Creator. However all of the steps taken in both applications are quite small and easily remedied by hand.
4) Consistency and standards
    a. Game Maker (and Game Creator as well) are only available on one platform and therefore only have one convention to follow.
5) Error prevention
    a. Game Maker seemed to have fairly robust error checking. I have yet to run across an actual error. This should be our goal with Game Creator as well.
6) Recognition rather than recall
    a. Because an object or character needs numerous properties, Game Maker divided them up into Sprites and Objects. To make up for this you can pick which of the other category you are working with from a drop down menu. While you do have to remember names, you don't need more then that.
7) Flexibility and efficiency of use

a. While these appear to exist within Game Maker, I am not at a point where I am using them. So I cannot say if they are useful or properly placed.

8) Aesthetic and minimalist design

a. This is part of the problem with using events rather then state charts. There are to many possible events the user might need to choose from. In addition you have all the conditionals to make sure the event only happens when you want it to. This is one of the aspects that will make Game Creator much easier to use then Game Maker.

9) Help users recognize, diagnose, and recover from errors

a. N/A. As I have yet to see an error message, I am not sure how helpful or not it is.

10) Help and documentation

a. Game Makers help system will walk you through simple problems but not much more. Of course there is not much more expected from a help system so if Game Creator can do that as well we will have succeeded.

Evaluation done by Ryan (3-5 small games made):

1) Visibility of system status

a. N/A

2) Match between system and the real world

a. Game Maker does stay close to reality in some respects but strays in others. They refer to imported images as sprites but then refer to levels as rooms. They also have no concept of menu's or credits so the user is left to turn a room into a menu. Game Creator tries to live a little closer to the way games are made in industry with the use of states. A menu is a menu state and a state with game play is considered an interactive state.

3) User control and freedom

a. The user has a fair amount of freedom in both applications. The user never feels as if they are on rails being taken through the development process. They can freely jump between creating objects (or actors) and levels (or game states) in both applications.

4) Consistency and standards

a. Nothing in either application strays from the norms of other development tools.

5) Error prevention

    a. Game Maker seems to be free of errors. Game Creator has some error states, but this is due to its stages in development and not because of its design. Game Creator would be as robust as Game Maker in its final stages.

6) Recognition rather than recall

    a. There is a fair amount of recalling names of user created elements in both applications, neither is unmanageable. Game Creator also takes things a step further by templatizing Menu, Dialog, and Interactive States while Game Maker relies on the user to create all the functionality for things like a Menu.

7) Flexibility and efficiency of use

    a. Game Maker does allow the user to create user defined scripts and actions for expert users and Game Creator will have the same options.

8) Aesthetic and minimalist design

    a. This is where Game Creator has a bit of an edge. Game Maker does not have any way to organize ideas when creating rules for objects. It is also quite cumbersome to create a simple menu. Game Creator streamlines both these processes through the use of states and by providing a way for the user to indicate whether a given Game State is to be used for a Menu, Dialog, etc.

9) Help users recognize, diagnose, and recover from errors

    a. I did not encounter any error dialogs with Game Maker. Game Maker does have debugging functionality but this was never used in any game I created using it.

10) Help and documentation

    a. Game Maker does have a standard help system but I did not need to use it. Also, Game Maker does provide a few example games but provides no comments that accompany the games. Game Maker is planned to come packaged with a few tutorials.

Evaluation done by Tawa (No prior experience):

1) Visibility of system status

a. N/A

2) Match between system and the real world

    a. The controls are setup with both domain specific and some program defined terminology. For instance, the use of *Sprites* in the add sprite command is logical with some experience with games and game terminology. However, the concept of a *Room* seems totally new and not very intuitive.

3) User control and freedom

    a. This seems very appropriate for the application. I can go in and out of various modes in a similar fashion to other IDE products I have used.

4) Consistency and standards

    a. This is exactly as expected. For instance, the use of the delete command is not only consistent within the application but also is familiar when compared to other similar development environments.

5) Error prevention

    a. Over the course of the evaluation, there were no errors.

6) Recognition rather than recall

    a. This is mixed. On the one hand, there are several commands, labels, and menu items that are easily recognized from previous experiences with other IDEs and similar products. On the other hand, a number of commands associated with the making new games were defined by the application and as such took some time to find and/or recognize immediately.

7) Flexibility and efficiency of use

    a. My understanding is that Game Maker is quite flexible. However, for the purposes of this evaluation, I was not concerned with this aspect. As far as efficiency of use goes, it is very good when compared to developing the game from scratch, and probably even when compared to similar libraries/APIs.

8) Aesthetic and minimalist design

    a. This was reasonable but could have been improved, I think. I think that the focus for the product was functionality rather than aesthetics of the design.

9) Help users recognize, diagnose, and recover from errors

a. N/A

10) Help and documentation

a. This was ok. However, on a whole I found that it was simply easier to figure out my problem by experiementing with the application rather than reading through several pages of documentation. I think, however, that is as much a testament to the overall usability of the product as it is a negative characteristic of the Help system.

Analysis of Heuristic Evaluation:

After looking at our evaluations it seems that Game Maker does a lot of things right, so if we can keep most of the core elements of Game Maker in Game Creator we should satisfy these heuristics. The only area Game Maker faltered in was in *Aesthetic and minimalist design* and I think Game Creators addition of state diagrams will help this area and produce a solid tool that will allow for easy game development from inexperienced users.

Expert Evaluation

For the expert evaluation, we consulted Zoe' Wood. She is an instructor at Cal Poly and a driving force behind the new game program here. We chose here as an expert because of her experience with teaching CSC 476, her role in developing the curriculum for freshmen students, and finally her role in developing the game program. Furthermore, Ryan's previous relations with her allows for open communication, which is important because of the current state of our own application.

We based our expert evaluation on the advice and outline provided by the following website: http://www.usabilitynet.org/tools/expertheuristic.htm

Here instead of focusing only on the evaluation of Game Maker, we chose to focus on the evaluation of our project (Game Creator) by demonstrating our current prototype to Zoe' and using Game Maker to fill in some of the gaps in our prototype in order to give her a more complete experience. That is, we walked her through a use case of creating a game with our product and then showed her how we would complete the rest of the interface. We also contrasted some aspects of our design and implementation to Game Maker. The major difference being the use of a State based development process (our product) VS. an event based development process.

The following is a paraphrased interview with Zoe Wood conducted on March 15th, 2007.

Interviewer: Do you think that Game Maker does an adequate job of allowing incoming freshman to create a simple game?

Expert: Their are a lot of things I like about Game Maker, but I am still considering Flash and other tools because I think Game Maker teaches very little about game development and is strictly a prototyping tool.

Interviewer: Does Game Creator look like it is addressing some of your concerns with Game Maker?

Expert: I really like the idea of Game Creator being centered around states. Most games have state machines at their core so a tool that can push students to think of games in terms of state machines sounds like a great idea.

Interviewer: Do you think state machines will be to advanced for some entry level students?

Expert: No, we will be teaching a class not just handing them a tool so we can help student grasp concepts like state machines.

Interviewer: Would you consider using Game Creator over Game Maker for an entry level game design class?

Expert: Of course, it sounds like all the core components that I liked about Game Maker will be in Game Creator with the only major addition being state machines. As long as Game Creator still allows for more advanced features that allow students to write there own scripts for actions, it sounds like a great tool.

Results and Analysis of Expert Evaluation

Our brief interview with Zoe confirmed that we were on the right track with Game Creator. She seemed to really like the concept of adding state machines to Game Maker and only saw benefits that could come from the addition. If more time was available our next step would be to conduct more interviews with students to see if they felt it would be easier to create a game with Game Creator since they had a way to better organize their ideas through the use of state machines.

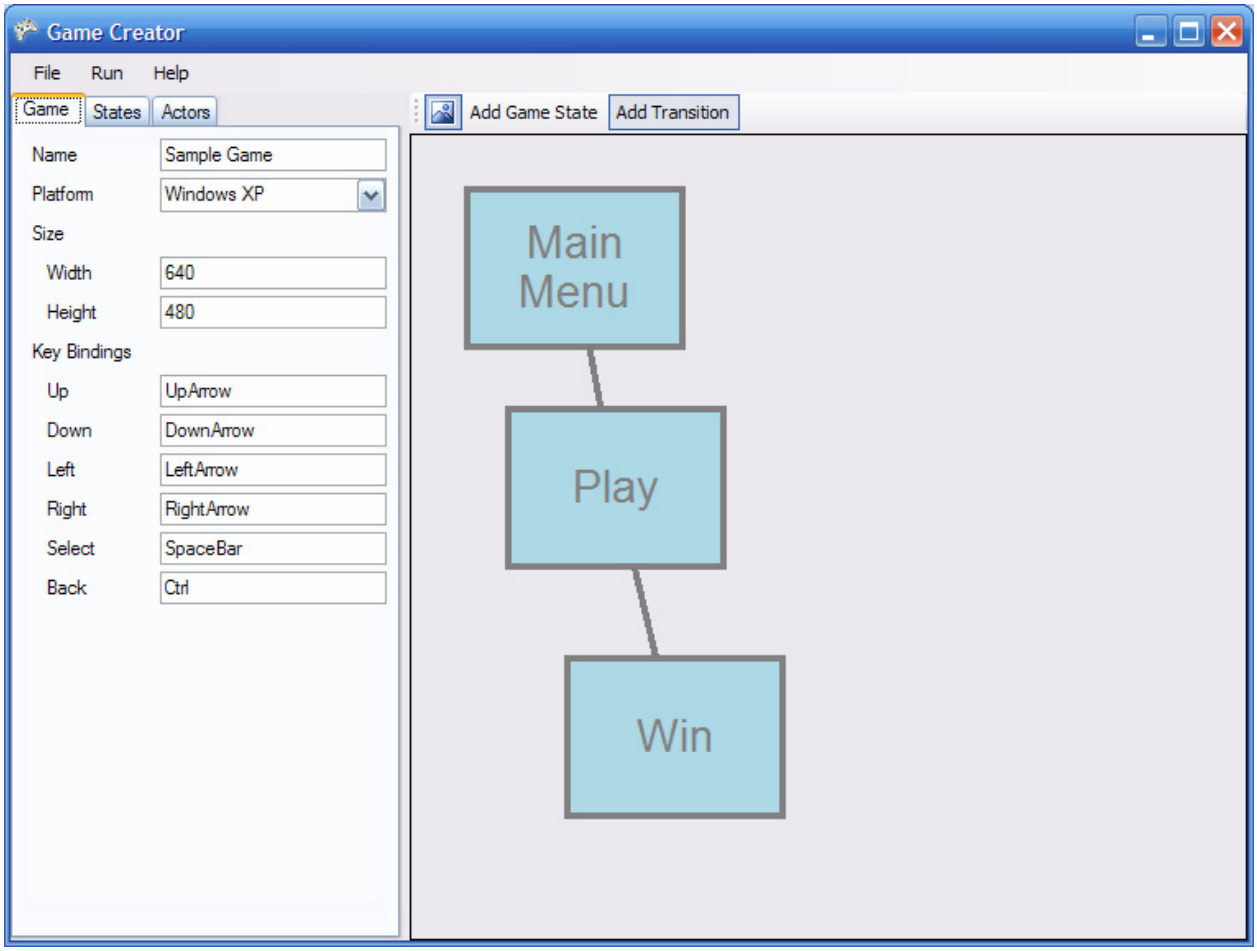
Game Creator Prototype

Our prototype has been developed in C# with the associated Visual Studio GUI builder and Widgit library. This in itself was a learning experience for us. This because although we have used C# for various projects, this is our first exposure to the GUI builder. On a whole, we have had a successful experience with the GUI builder.

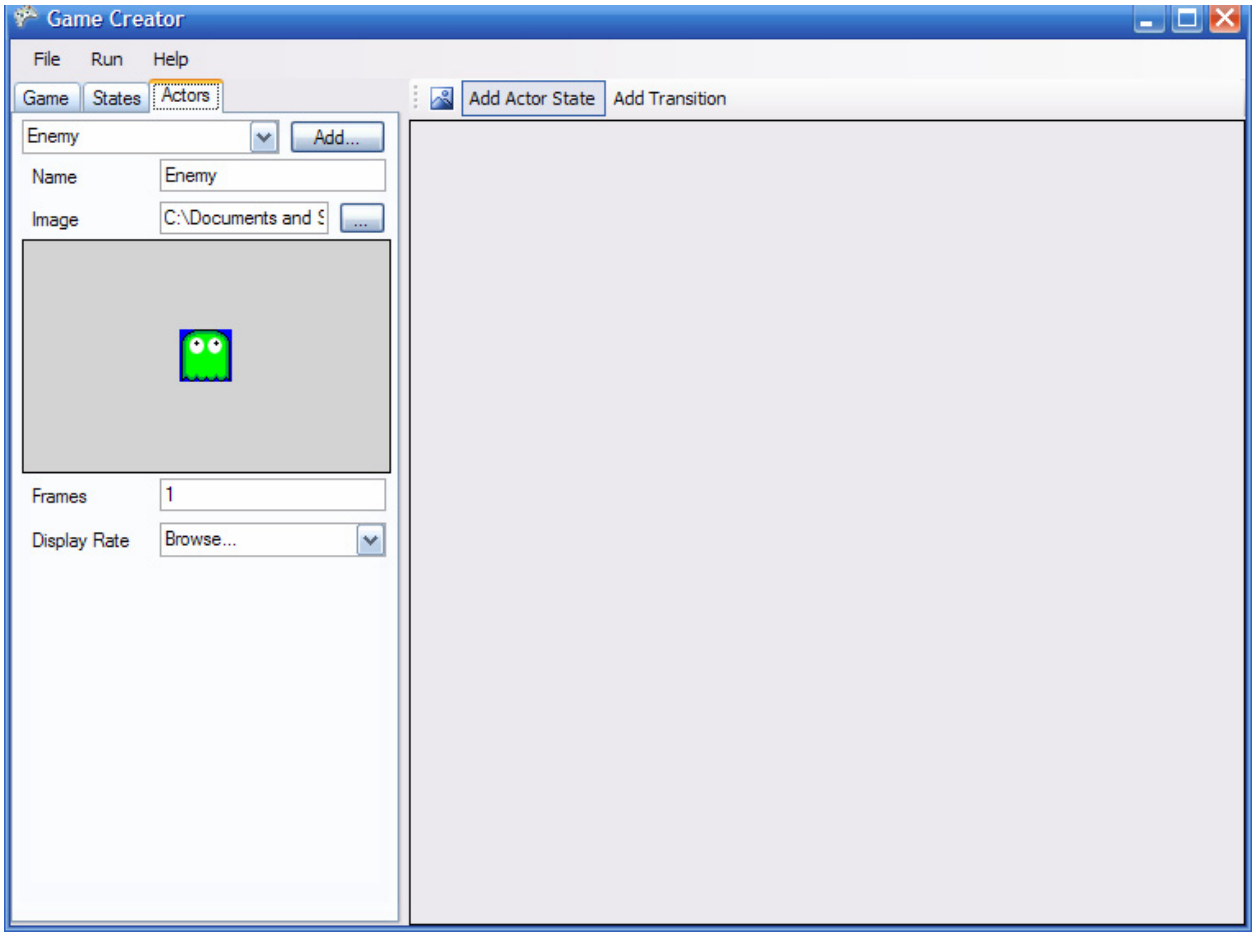
This a more functional version of the product that we first proposed for our storyboard (see the *Game Creator Overview* section above). The main differences is that we have implemented much of the functionality that we previously photoshoped in. For example, we are now able to add Game States, Actors, Game Background, and various properties of the Game States and Actors. Furthermore, we were able to implement transitions between the Game States. However, we found the transitions between Actor States were more complicated than we expected and due to time constraints, we were not able to complete these this quarter. On the other hand, we are confident that with more time, we can solve this problem.

The following is a series of screen shots from our current prototype. These follow a similar outline to our storyboard:

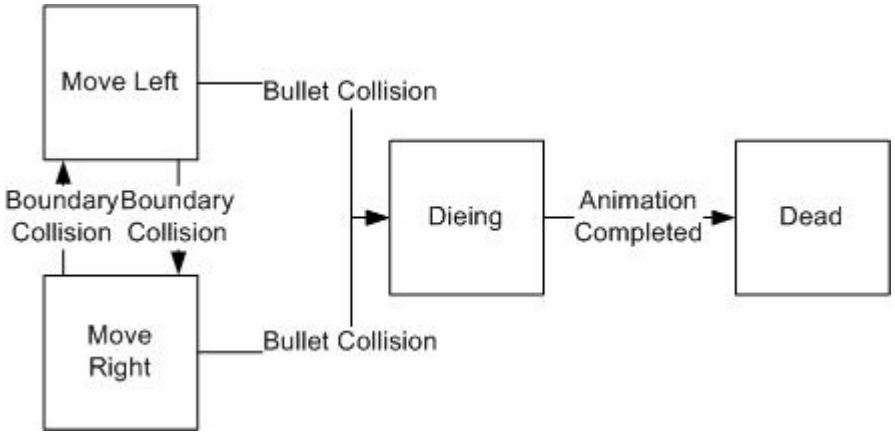
Here is the main screen with several game states added (*Main Menu, Play, Win*) and the transitions between them. The flow of transitions is from the top (*Main Menu*) and goes down.



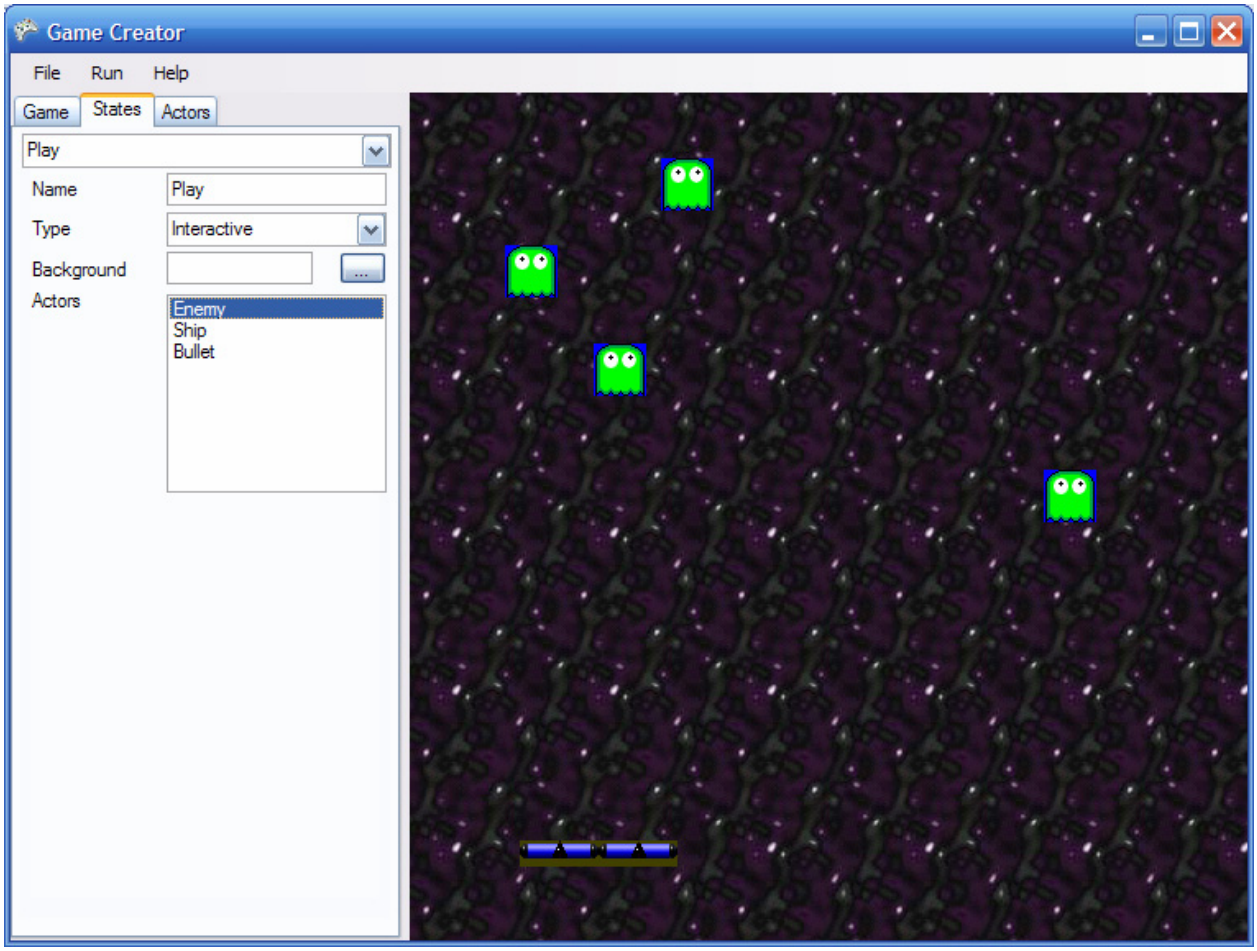
Here is a shot of our Actors screen with an Enemy actor created.



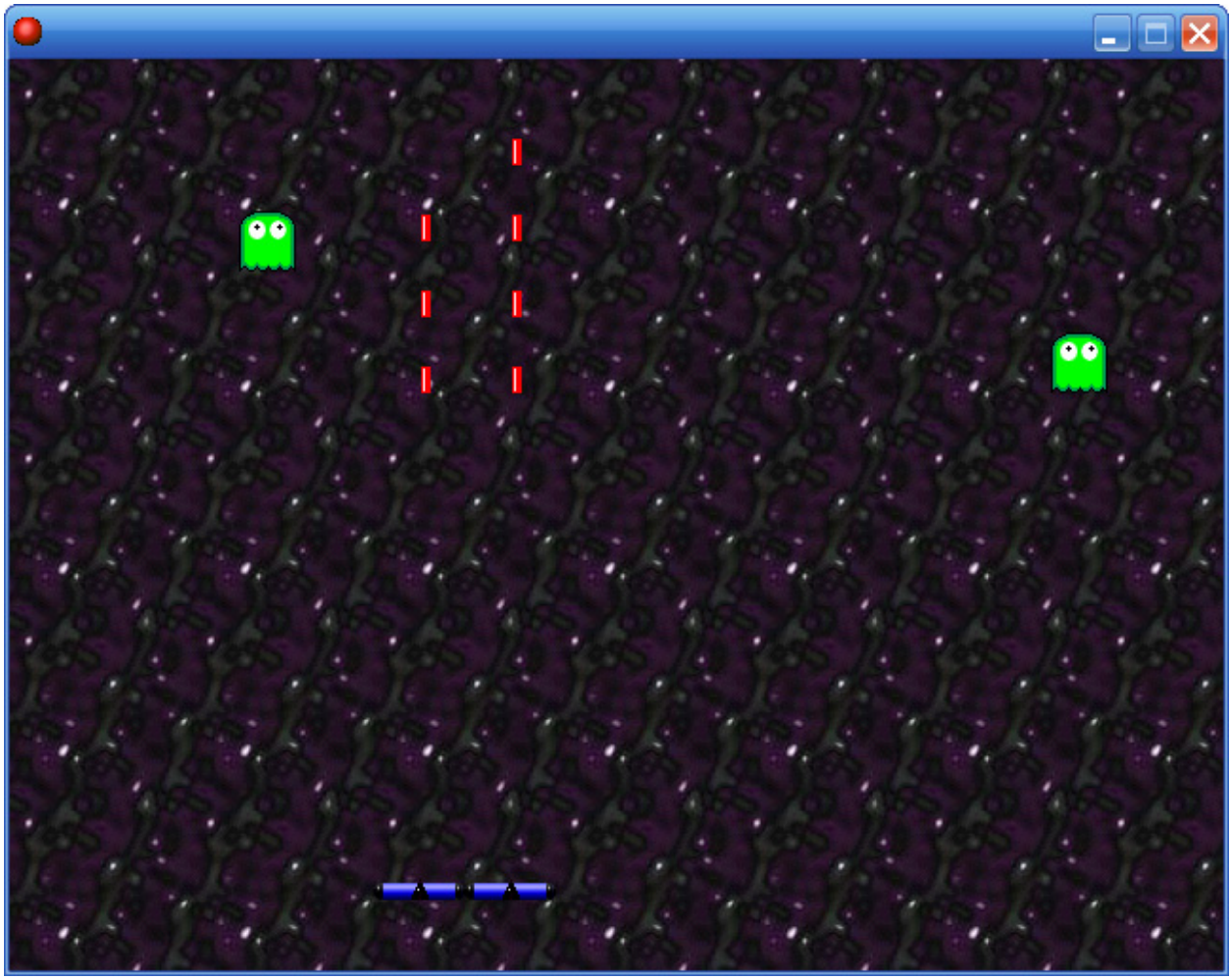
This is essentially the functionality that we are missing at this point. That is, we still need to implement the ability to add more complex transitions between states. In the future, we will have the ability to make diagrams similar to this one within the application.



Here is where we set up the background a place the initial location of the Actors. This specifies how the game will start once the user selects the *Play* option from the Main Menu state.



Finally, here is an example of what the game would look like once we are done.



Conclusion

We are pleased with our results for this quarter. We think that our prototype has promise and look forward to implementing more of it in the future. Furthermore, we believe that the use of states as a conceptual model provides an excellent experience for making simple games for people who have little exposure to programming and would rather not be bogged down by the coding side of game development. Additionally, we hope that such an interface will provide a more advanced developer with an easy way to build simple prototypes for games.

With this said, however, we understand that more work is required to fully implement Game Creator, and that more evaluation of the finished product will be useful, especially with regard to the state based development model. On the other hand, we have some advantages because much of what we are doing is just an extension of other products such as Game Maker and other IDEs.

Future Work

To continue this product, we would like to do the following:

1) Implement more complex transitions for Actors. The transitions would be complex enough to model multidirectional transitions (i.e., arrows rather than just top to bottom or left to right) and allow us to associate a specific event (label) to the transition.
2) The above would only apply to the GUI (frontend), and from there, we want to add the backend. We envision this to be essentially an XML file that then can be parsed and translated/compiled for different platforms (i.e., Windows, Brew, J2ME, etc).
3) Finally, we would still like to do more research with students.
    a. Since the results of the first survey were inconclusive, we would like to run it again. This time with either the prototype Game Creator or the finished project. Given the opportunity to actually create a game off these state charts, we believe that we will get more interest then the previous survey.
    b. We would like to run a pilot test with an actual 100 level course and our product. Thus the new 100 level game creation course is started.

References

[1] http://www.gamemaker.nl/

[2] http://projectfun.digipen.edu/ProjectFun/index.aspx