# Experience in the Game Design Seminar

Adam Smith (amsmith@cs.ucsc.edu)
Spring Quarter 2006

## Introduction

In UC Santa Cruz's first ever offering of a Game Design Seminar, I had the involving experience of developing four games from scratch, participating in discussions relevant to the times, and developing my skill in computer game design. This paper documents my thinking, efforts, and analysis of mine and other's games developed for the seminar. In the first section, I will briefly discuss the mechanics of the seminar. In the second, I will describe a model for the analysis of games that I developed the quarter before this seminar. Next, in the third, I will take an in-depth look at my own games, analyzing them under my game model, and discussing their strengths and weaknesses. In the fourth section I will apply the same analysis to a sampling of games written by other students. Finally, I will look at the effectiveness of the themes provided in the seminar as a guide for game design and some alternative phrasings of the given themes that expressed themselves in the games developed this quarter.

## Overview of the Seminar

The Game Design Seminar, recorded as Independent Study on the record, was a 5-credit (standard weight) class targeted at those students who had completed the Introduction to Game Design class the quarter before. Extensive programming and media creation experience was not required, however the varying skill-sets among students was visible and lent a wide diversity to the look and feel of the games developed. The seminar met weekly for two hours during which class time was split between discussion and demonstration of student games. Discussion topics included the effectiveness of different input devices, analysis of the popularity of platform jumping games, reactions to technology demonstrated at E3, and the importance of story to creating involving games (among many other topics). During demonstrations, the student developer was given roughly ten minutes to show off the game-play of their creation as well as discuss any motivations or development issues that were not visually evident. Other students were given a chance to play each game while discussion of the strengths and weaknesses of the game, as well as suggestions for further development continued. Most students provided written feedback to the presenter in the form of a small paper slip with a few numeric options and a free-form response section. This free-form response was often the most helpful to gaining insight as to what people really thought of a game. It is worth mentioning that the class was split into two teams. Blue team games were presented a week after red team games as a way to give more time to allow students to present each meeting and allow for discussion to be mixed into each meeting. Only within optional groups did students work together. The colored team members shared the common experience of having been exposed to the same inspirations from the

week before. A new game was due each two-week interval and students were heavily encouraged to try to build a given theme (a simple word) into their games. Students had mixed reactions to the themes. My reaction is described in the last section of this paper.

I should point out, to clarify my position amongst the other students, that I was one of two computer science graduate students in the seminar. I had previous computer game design experience (mostly focused on web based games) and significant media creation experience from personal projects. My perspective on game design was most heavily shaped by my recent experience as the teaching assistant for the Scientific Visualization, Computer Animation, and Games class the quarter before, where I taught students my vocabulary (which I describe in the next section) for talking about the structure of games and guided them through the development of one, and, in some cases, two, physically-based, real-time, 3D games in the space of a few weeks.

## Game Model

In my model for games, it is important distinguish four independent components. These are rules, mechanics, inputs, and outputs. This model was designed to guide discussion of real-time, 3D computer games, however it is applicable to many types of games. As such, the model implicitly deals with games having some notion of game state and time. Events are also involved in this model simply as a statement about action at a certain time. The complex connections between the game state, time, and player(s) are what these four components describe. I attempt to directly define, in clear language, what each of these components are and leave showing examples of them to subsequent sections.

Rules, in this model, refer to a collection logic that defines a mapping of game state and events to new game events. Events spawned by the rules may represent victory or loss, thus encoding goals or objectives of the game. Alternatively, rules may define the capability of the player to take and action at the next turn. This capability can be thought of as an event nonetheless because it may either be present or not at certain time steps. Rules can be applied in either a continuous or discrete fashion if considered to be general event-handlers, where some rules respond to a per-frame tick event.

Mechanics refer to logic that defines a mapping of game state and events to new game states. In the absence of any events (or the presence of only the tick event) the mechanics directly encode the dynamics of the game world. With events present, the mechanics define how intended actions are actually carried out. Because rules and mechanics are both a function of game state and events, and each component produces the next round of each game state and events, the dividing line between these two components is sometimes hard to find and may be arbitrary. Mechanics may attempt to follow a strict physical model or encode a more-fun-than-realistic notion of how the abstract game world works.

Inputs refer to the means by which the player creates events in the game world. By spawning events with input the player is able (through mediation by rules and mechanics) to attempt changing the game state to their liking.

Importantly inputs do not involve the game-state at all (beyond what is indirectly influenced by the rules). In the realm of computer games this maps to discussion of to what the keyboard, mouse, joysticks and any other input devices are logically hooked. Aesthetics aside, it is usually desirable to provide the player with fairly direct input to the aspects of the game world that it makes sense to control. However, even cheat codes and game configuration options can be discussed as inputs of a game.

Finally, outputs refer to the means by which the game state is communicated to the player. Outputs deal only with game state and not game events. If it seems that game events are being represented in the outputs of the game, it might be useful to consider the presence of that event a part of the game state for consistency. Usually, this refers to how what aspects and how effectively the game state is represented via the computer's output devices (audio, video, force-feedback). Again, aesthetics aside, it is usually desirable to have a direct connection between the part of the game state that the player should intuitively have access to and the outputs provided by the game. Often, this means that there is much of the game state that the user cannot see and must observe indirectly.

As a disclaimer, the terminology I just introduced will be used heavily in the subsequent sections as a way of talking about the individual parts of a game, ignoring holistic properties of a game. Often, however, it is possible to trace the strengths and weaknesses of the design of a game to one of the components. There are, obviously, other ways to break down the design of a game.

Additionally, an important component, which I will adopt for discussion (although it does not clearly fit into the above model), is the notion of a core mechanic. I take the core mechanic of a game to refer to the simple, general action that describes what the player is doing while playing the bulk of the game. In my model, this is an aspect that cuts across the components of input and mechanics.

As a final note, it is often easier to phrase the rules and mechanics of a game as the sort of thing one would find in the rules section of the manual for a board game. The more mathematical definition of rules and mechanics maps well implementation of a game but is slightly inconvenient when it comes time to enumerate all of the possible events and state variables required to define the game. In further discussion I will use this more rulebook-like approach.

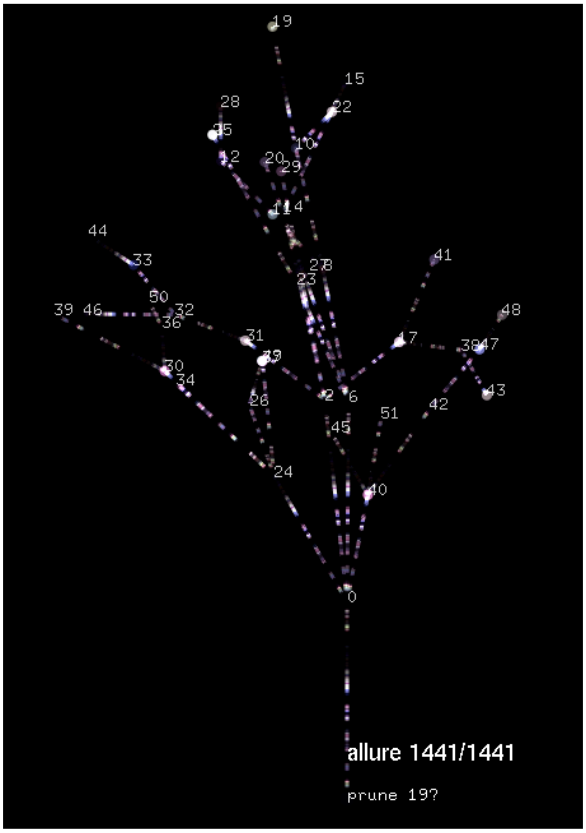
## My Games

In this section I will look at the details of the games I created. For each game I will provide a screenshot and simple description, my assessment of how the theme for the game was tied into its design, an analysis of the game under my model, analyze the success of the game (tying back to the game model where possible), and, finally, add any reactions I had to the final result. Before I go on with the per-game description there are certain crosscutting motivations I had that are worth mentioning.

In every game I made, I tried, at least initially, to build the given theme into the core mechanic. That is, I tried to make the given theme be the thing that

the player actually does in the game, and not only tie it in superficially. This was consistently a difficult design task. Furthermore, I aimed to do something very non-obvious, often tying the game back to mathematics or computer science at a deep level.

## *the.discrete.gardener*




This game was a real-time, 3D, growth simulation. During game-play, the player prunes a slowly growing abstract tree in attempt to improve its "allure." Depending on how often the player decides to take an action the game has either an action or strategy feel. Evolving procedural visuals and original electronic music accompany the core game-play and provide futuristic, hacking feel to a game that would otherwise evoke a more traditional, bonsai tending feel. The title was picked mainly to allow the play on words for "discrete" and to create contrast between the image evoke by the title for those who had seen the game and its title spelled out and those who had only heard its title. It was programmed in Python using OpenGL and the game template I developed in C++ as a teaching assistant for use in the class the previous quarter.

The theme for this game was evolution. I turned to the dictionary to find an interesting, alternative definition for this word. A secondary definition of evolution, devoid of references to this history of life on the earth, involves the gradual development of something from a simple to more complex form. In my game, this is expressed as the tree slowly growing more complex, hence evolving, over time. The inclusion of pruning and the fitness measure of the tree allude to the primary definition of evolution. By careful selection (intelligent design!) the player can guide the development of the tree in a somewhat natural way to their desired configuration.

In term of my game model, I will analyze this game in more depth than others to give a clear example of decomposition into the four components.

Rules: Player may prune any node in the current tree but the root. Pruning a node removes its children from the tree. The tree grows with every fourth beat of the music.

Mechanics: Allure is calculated as the quotient of the number of nodes in the tree and the size of the tree (this builds in a preference for complexity and compactness). When a new node is created its parameters are inherited from its

parent node with some mutation (this is a direct tie-in to the evolution theme). There is an equal probability of any node growing next (creating leaf-heavy trees).

Input: The player may user the keyboard digits are used to enter numeric identifier of node to be pruned. In this sense, the partially typed node identifier is part of the game state. Hitting the Enter key attempts a prune action with the current identifier.

Output: The tree is rendered in 3D. It continuously rotates to better show its structure. Music loops to indicate continuous nature of game. Noises play to confirm keystrokes and prune operations.

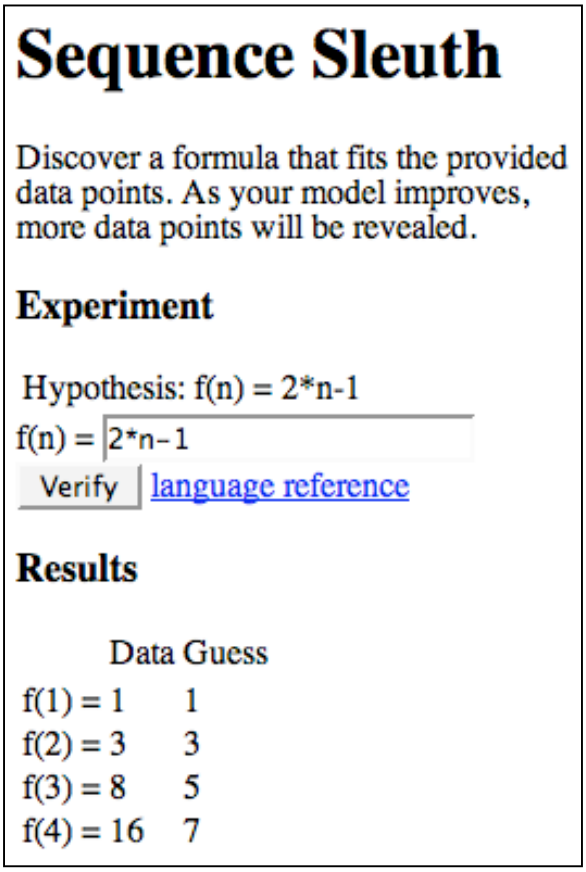
Overall, I feel that this game was very successful. There are several reasons for this: The mechanics were simple, the rules were not unnecessarily restrictive, the input was fairly direct (however clicking with the mouse might have been easier), and the output was a clear representation of the game state. On a holistic level, the "feel" of the game was consistent and reinforced by the input and output components of the game. One general shortfall of the game was that it was not accessible to everyone. Its dynamics and goals were highly abstract and had little connection to real world activities (although this only strengthened the computer science connection).

I was surprised by the final balance of time that went into this project. Roughly 40% of the time was spent creating media (art, music, tuning animation parameter), 40% was spent porting the game template to Python for faster development, and only 20% of the time was spent programming the solid rules, mechanics, input and output of the game.

## *Sequence Sleuth*

This game was a web-based, handheld-friendly puzzle of function fitting. The player is presented with a few data points and a sample hypothesis, *f(n)=0*. From here, the player must iteratively guess new functions that might fit the data and test them to progress through the puzzle. As the hypothesis becomes better at fitting the data points, new points are exposed. This game was developed in PHP, outputting XHTML 1.0 Strict for maximum device compliance.

Discovery was the theme for this game, and, again, I tried to build this into the core mechanic of the game. During game-play the



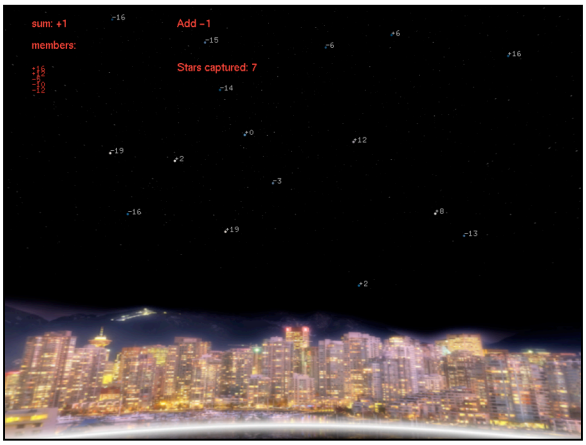
player is actively discovering how well various hypothesis fit the given data points. The cubic and quadratic terms of the sequence generator are only visible after several terms have been fit, ensuring that it is very unlikely that the user would discover the best formula early on.

As this game is far from the real-time, 3D simulation that my model was developed around, its description is fairly simple in terms of the four components. Rules: A new hypothesis triggers a reevaluation of the player's guesses. A flawless hypothesis triggers the victory message. Mechanics: A hypothesis is evaluated by comparing the values generated by the player's expression to the puzzle's hidden expression. New data points are revealed until the player's hypothesis was violated more than some threshold number of times. At this point the game sleeps until a new hypothesis is proposed. Input: The player inputs their hypothesis, in a simplified PHP syntax (sigils automatically added), into a web form. An interesting side effect of this is that the game state can be trivially saved using a browser's bookmark feature. Output: A table of data values and predictions is displayed on a web page.

As described, the game was successful. I was unable to demonstrate play over the network in class, however, there is nothing limiting this ability. The game was extremely simple in its aim for core mechanic and hit that mark. At a holistic level, however, the game was somewhat disappointing having no enticing media or story. In this sense, it is entirely suitable to the devices for which I targeted the game. More so than the previous game, the game play was inaccessible to many, especially those with an ingrained impatience for mathematics.

My original vision for the game involved a system by which puzzles could be simply created and sent to friends with a simple link. Furthermore, I wanted to add additional mechanics that would keep track of the number of iterations required to reach a flawless hypothesis. I even considered rendering colorful plots of the player and data functions. This, however, would not have changed the game play much.

## *I Loves Me Sum*




At its core, this game is a search-based puzzle, however, due to changing aim several times it feels much more like a real-time, 3D whack-a-mole game. The goal of the game is to collect stars from the night sky. The player accomplishes this by activating sets of stars with values summing to zero. This game uses the Python game template I created for the first game and some hand-created but not-for-this-project art and original ambient music. The feel and mood of this game are somewhat spoiled as input and output for this

game were borrowed from the first, evoking memories of a very different game at the core.

The theme for this game was love. In contrast to the rest of the games I tried to make what player does in the game not be described as love (the verb), although I later went back on this aim. In the story (only presented verbally, not in the code) the player is collecting stars to give as a gift from a romantic skyline on a clear night. The screen was purposely made to be too visually busy for one person to handle, to encourage play with a loved one. Finally, as a last campy hack, the core mechanic was tied back to love (without being love related at all) by focusing on a score or value of zero in the tennis sense.

In terms of my game model, this game has a rather complex, though artificial description. Rules: Only existing stars may be added to the active set. If the stars in the active set have values summing to zero, they are all collected. A new star is added every fourth beat of the music (this was a mistake).

Mechanics: When a new star is added, it is assigned a random integer value within a fixed range (this was a killer mistake). When stars are collected they no longer exist and their count (not value) is added to the collected stars score. Input: Keyboard keys are used to input a star's (non-unique) value. Depending on mode the identified star is added or removed from the active set. Output: The contents of the active set are displayed in text along with the running sum of the set's members. A rotating, 3D night sky shows randomly distributed stars (some stars are too high for the camera to see, unfortunately). Music and noises are present in similar effect as in the first game.

This game was clearly unsuccessful. The (quite artificial and unintuitive) rule that created a new star at regular intervals, while making the sky very busy as intended, made it too easy for the player to enter numbers at random without looking at the sky to do any searching. Additionally, the mechanic that assigned values to new stars at random made confused the game play, as active set was a true set while the sky was a multi-set. The input scheme was certainly not optimal, though the player had quite a few more possible actions in this game than the evolution game. In contrast to the discovery game, all that this game really had of value was its feel (afforded by its output).

The motivation for this project was a bit of advice I gave out but never took myself the quarter before. I told my students that if they needed a game idea they could simply go find a list of NP-Complete programs and easily adapt any of them into a puzzle game that, despite our researchers best efforts, have no simple solution. While I had several examples of games ideas that followed this pattern, I had trouble adapting even the subset-sum problem to the love theme. This was primarily an issue of creativity and I am still confident that there are many NP-Complete problems that can easily be adapted to puzzle games in the case of decision problems and time-based or high-score-based action or strategy games in the case of optimization problems.

## *RainMakr*

This game was a 2D, real-time game of transmutation. The player manipulates a cloudy sky using a tool that attracts clouds and converts them to rain. This rain, hopefully, falls on plants that convert it into health and new growth. By really only manipulating the interaction of cloud puffs the player is able to manage the well being of several simulated plants. This game was programmed in Processing, a Java-based environment for creating simple interactive visual programs (similar to but in different aim than Flash). Before this project, I was unfamiliar with development in Processing and this style of development in general.

For the tie-in to the theme, water, I originally planned the game being about the player watering some plants. As I discovered more of the graphical capability of Processing the focus shifted to showing off the various water-looking particle systems I had created. This still, of course, ties into the theme, but in distinctly different way.

This game was somewhat incomplete, and this shows up in the decomposition into my model. Rules: When a cloud touches the tool it turns to rain. When rain touches a plant it waters the plant. Victory and loss condition rules are notably missing. Mechanics: Clouds drift with gravity-like attraction towards the tool. Rain falls downward with predictable constant acceleration. When a plant is watered its water storage, health, and size are updated according to a complex (difficult to tune) system of differential equations. The tool moves directly to the motion target each frame. Input: The player creates a motion target for the tool by moving the mouse. This is the player's only input (intentionally). Output: Fluffy clouds and rain chunks indicate the position of point-particles in the game state. The size and color of a plant represent its abstract size and health respectively. The water storage part of the plants' state is not represented to give the user something to guess at and learn by experience. The tool is represented by a Firefox logo.

This game could have been quite fun if it was completed. Additional rules, multiple levels, high scores stored on the web all would have made this game more fun. Processing applets are well suited to added web back-ends. However, I feel it is a good example of a successful demonstration of the core

mechanic and a complex output component. Iterative development like this was sorely underrepresented in the seminar, I believe.

I regret spending so much time experimenting with Processing that I lost sight of a complete game. The Firefox logo, an arbitrary placeholder, while cute, made very clear the incomplete feel of this game. On the brighter side, not emphasizing the growth model of the plants and focusing on the transmutation aspect instead made this game more accessible to those who I had turned off with my mathematical games.

## Games by Other Students

In this section I will follow a similar thread of discussion to the previous section but explore the details of some of the games created by other students. I will focus on three games from the first theme presented to the seminar, evolution, as an example.

### *Populution*

This was a 2D game of cellular-automata based population growth simulation. It was developed in Java in a group of two people. Its game-play follows a guess-and-check pattern for improving the parameters of competing civilizations to create interesting growth patterns.

My game model describes simulations without trouble and this game has a clear decomposition. Rules: Player announces civilization parameters and starts simulation. Mechanics: Fairly complex, parameterized cellular-automata rules define the state of the world directly for the next time step. This state, in addition to grid-space values, included overall values representing quantities like the level of development in technology, medicine and culture. The simulation runs until terminated without further input. Input: The player edits XML configuration files in their favorite text editor. Output: pixels are mapped to grid cells and colored based on population density. Overall values are displayed as text. Other grid-space values like resources and local development are not displayed.

While, by some models, this is strictly not a game I still quite enjoyed the demonstration of this project. The theme was clearly represented in several aspects of the simulation, both in the increasing complexity of the competing civilization as well as the iterative genetic improvement in the meta-game of hand editing the configuration files between runs. I would consider this game very successful despite its lack of a clear, holistic feel. The simple story of the world (civilizations with certain attitudes and intentions) was cute and motivated inventing narratives to go along with the evolving visual representation of the world.

### *Darwin*

This comedic but simple game was a variant of the simple warmer-colder game played with children. The player, represented by the sprite of a monkey with a drawing of Charles Darwin for a head, waters around a colorful terrain including trails, cliffs, and a realistic water surface simulation looking for a particular creature. The world is full of strange combinations of real creatures

(Photoshop products, certainly) that inform the player whether they are getting closer or farther way from the target creature. The creatures speak with silly voices and sometimes give nonsensical reports.

This game is straightforward in terms of my model. Rules: If the player is in range of a creature they have never spoken with and the user wants to speak, have that creature report the distance using knowledge of the location of the last creature you talked to (and a very complex path finding system). If the player attempts to speak with the target creature, have that creature report a disappointing victory message. Mechanics: Darwin can walk around the terrain (sinking in water) with simple vehicle navigation. The surface of water is simulated according to common equations for this application (completely unimportant to game-play). Input: Standard first-person shooter controls map to Darwin's movement. Holding down a certain key will attempt talking to a creature (successful if on is in range, according to the rules). Output: An opaque height field is rendered to represent the terrain. Other Creatures are simple billboard sprites like Darwin and become tinted green when in range and tinted red after they have been spoken with. Original MIDI music plays in the background, representing that, indeed, time is moving forward.

In my estimation this simple game was successful, however the complexity of underlying simulation only serves to poke fun at the simplicity of the game play. Some graphical bugs were evident, but none that affected game-play. The vocals (original) and art (borrowed) went a long way to creating a feeling that was clearly described as fun. The theme was tied in a story element only, but did so in a clean, complete manner, so this is just more evidence of success.

## *Feathuckers*

This game was a simple, 2D, Java game that attacked the theme directly. During game-play the player, a pellet, chases down other pellets that have varying characteristics (genes) that determine their reaction to the player's pellet's relative distance and distance to the edge of the screen. The player must eat (run over) half of the pellets in a level to complete it, at which time the next generation of pellets is prepared from the genes of the surviving pellets. The games difficult progressively went up in direct response to the player's action and not a predefined schedule.

This game was another clear case for my model. Rules: Player must eat half of pellets to clear level. Pellets choose action based on their genes. The Player warps around the sides of the screens (toroidal topology). Mechanics: All pellets are subject to momentum, drag, and their navigation force. New genes are created by mutation from survivors of previous generation. Input: Arrow keys control the navigation force on the player's pellet. Output: Pellets are drawn as simple outlined circles on a flat colored, square field. Level number and progress are indicated in text.

This had a clear aim for putting evolution in the core mechanic and achieved this goal. The holistic feel of the game was somewhat bland, meaning this game probably is less impressive than others in terms of screenshots, but its

more playable than others. The dynamic, procedural nature of the game is impressive from a technical standpoint, but it probably lost on those who only see it as a simple chasing game that eventually gets too hard. Nonetheless, it is another successful game.

## Themes as Guides

In the games demonstrated in the seminar, I observed several ways to apply the given themes to a game's design. The different ways can be broken down, roughly, by the choice of the part of speech attributed to the theme. In class experience, which I will present below, verbs match nicely with integration into the core mechanic. Adjectives and adverbs are easy to use to flavor the media and feel of a game. Finally, nouns often easily map to story elements such as setting and plot. The themes given in the seminar were all presented as nouns so its not surprising so many games (other than the ones I previously discussed) included the theme as a story element. As guides, however, the themes could have been more effective with only simple changes. In general, I think presenting the themes only as nouns, or without a specific sense in mind left too much freedom for students to interpret the theme, lessening the amount of experience they would have shared solving the same problem. If themes had been presented with a clear sense or application, I feel they would have been better guides.

Consider the examples: "smash (verb) – core mechanic," "icy (adjective) – feel," and "guitar (noun) – story element." These all have a clear guide for what to do (though calling them themes may be less appropriate now). Below, I propose some alterations to the given themes and how these new themes were already expressed interestingly in the games demonstrated in the seminar.

Evolution: evolve/grow(v), evolutionary(adj), evolution/growth(n)
In the *the.discrete.gardener*, I represented the idea of evolutionary(adj) in the visual representation that grew more complex, despite perfect play being possible with a less complex display. In *Feathuckers*, evolve(v) was directly involved in the as the player caused the other pellets to evolve in the genetic sense. Finally, in *Darwin*, evolution(n) was added as a story element without touching the other words listed above.

Discovery: discover/invent(v), discovery/development(n)
In *CyBuilder*, invent(v) was integrated into the core mechanic as player invented novel combat robots by combining a wide array of parts together. In *PING.EXE*, discovery(n) was used as a story element as the civilzation of Mary's New Laptop being exposed to the wild world if the Internet for the first time in their history. No adjective variations for discovery were listed above because I found it difficult to thinking of a clean mapping of the idea of discovery to an adjective, especially one that could be applied to shaping the media or feel of a game, that was also demonstrated in class.

Love (a challenging theme): build/tighten/adapt/collect/yearn(v), warm/cozy/infinite/idealized/romantic(adj), excitement/mystery/romance/relationship/sex/gifts/sacrifice(n)
In *I loves Me Sum*, I attacked collect(v), romantic(adj), and gifts(n), and missed on most counts. In *Diamond Solitaire*, romantic(adj) was used in the art and

media as sparkling diamonds for puzzle tokens. Finally, in *Cupid's Conundrum*, relationship(n) is used in the story as people loved or hated others in response to arrows from the good or evil Cupids.

Water: hose/pour/dilute(v), watery/wet/cool(adj), water/liquid/ocean(n) In *Fire*, hose(v) was used in the core mechanic as player pushed around and soaked many different object. In *RainMakr*, the core mechanic was not water(v) despite it being a component of the game. I centered the story around water(n), leaving transmute(v) as the core mechanic. In *Jack the Basilisk Lizard* (a platform game), watery(adj) was used in art as level and enemies had watery components (but could fly/walk nonetheless). Very clearly, in *Deep Descent*, ocean(n) was used in the story as a submarine saves world by disarming underwater bombs.

## Final Thoughts

The class was composed of discussion, demonstration, and (for the graduate students) a critical analysis. These elements, combined, had a very significant contribution to my experience of game design. The in-class discussion of games opened my eyes to the many levels at which games can be understood and enjoyed. This discussion applied to my own games showed me that there were significant levels at which the majority of the audience was not interesting in my game. In my critical analysis and experience with others demonstrations, I realized my model was completely missing the vocabulary to talk about the holistic feel of a game, which is extremely important in the enjoyment of a game from a non-technical standpoint. In comparing the discussion of individual games with the discussion of general game design topics I realized people are much happier talking about what they know and are comfortable expressing solid opinion about. Despite having a very different background than most of the students in the seminar, I feel I was still able to take away a wealth of knowledge and new experience.