Software Prototype/Proposal Document (SPD)

22 September 2010
Version 1.0

Team Snow Crash
Dale Earnest
Mike McWilliams
Jeff Dunn
Dong Luo

Table of Contents

Revision History

Name	Date	Reason For Changes	Version(s)
Dale	20 Sept	Initial version	0.1
Dale	21 Sept	Added mockups, hours, edited for grammar and content	0.2
Dale	22 Sept	Final version	1.0

1.Project Proposal

1.1Summary and Purpose

This project will create an evolution simulation program. This Java-based desktop application will teach basic evolutionary principles in a fun and engaging manner.

1.2Description

The user of the evolutionary simulation creates (or uses default) “critters” that inhabit a simple game world. Critters inhabit a game world and strive against each other in competition for resources and space. Successful critters reproduce, pass their genes, and thrive while unsuccessful critters die out. They have traits that make it easier, or more difficult, to survive in this world and pass on their genes to future generations. After a set number of turns, the results of the critters' interactions in the world are evaluated and the user is presented with statistics on how the world has changed.

1.3Basic Design

The user may generate critters or use default critters to populate a simple game world (a flat 2D bounded grid). Simulations run for a set amount of time.

Critters are Plants (that thrive over time), Prey (that eat Plants), and Predators (that eat Prey). Each type of critter has a series of pairs traits that it can pass on through reproduction. Traits could include:

- Vision distance
- Camouflage
- Claws
- Movement speed
- Endurance
- Size
- Age

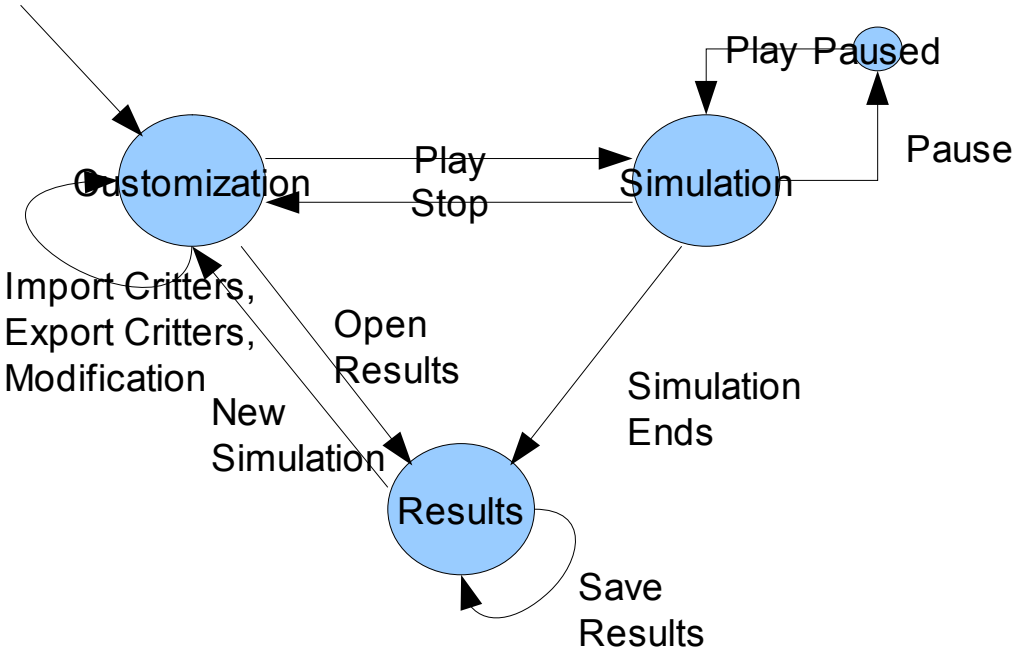
Each critter is built using a number of points. These points determine how many traits can be applied to the critter. For a balanced simulation, all critters should be built with the same number of points. For an unbalanced simulations, critters can be built with different numbers of points.

Critters are randomly placed on the map at simulation start. Only one critter may occupy a space on the grid at one time. Plants do not move. Prey and Predators move according to their type.

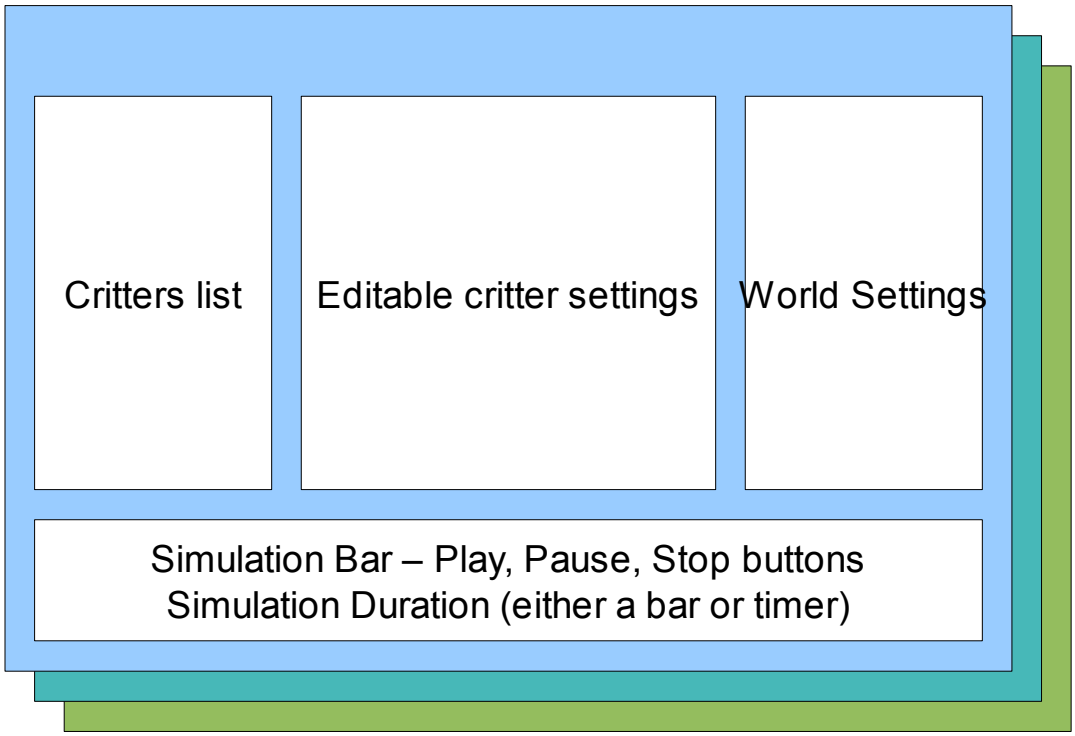
Each creature has a “health” meter; when the creature eats, the health increases (up to some maximum). When a creature doesn't eat, the creature's heath meter decreases every round until the meter reaches zero (death). All critters disappear after death. If a critter eats another critter, then the “eater” replaces the “eaten” on the grid.

1.4Look and Feel

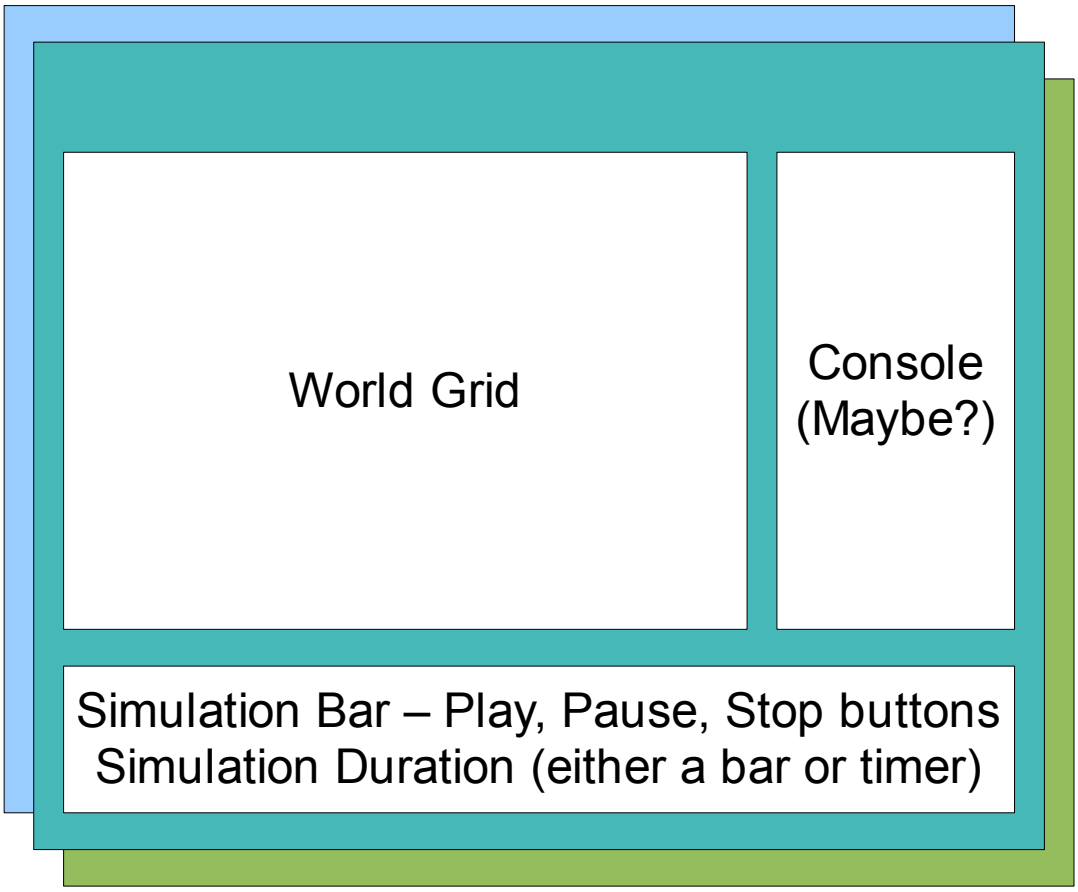
1.4.1Proposed UI Screen Flow



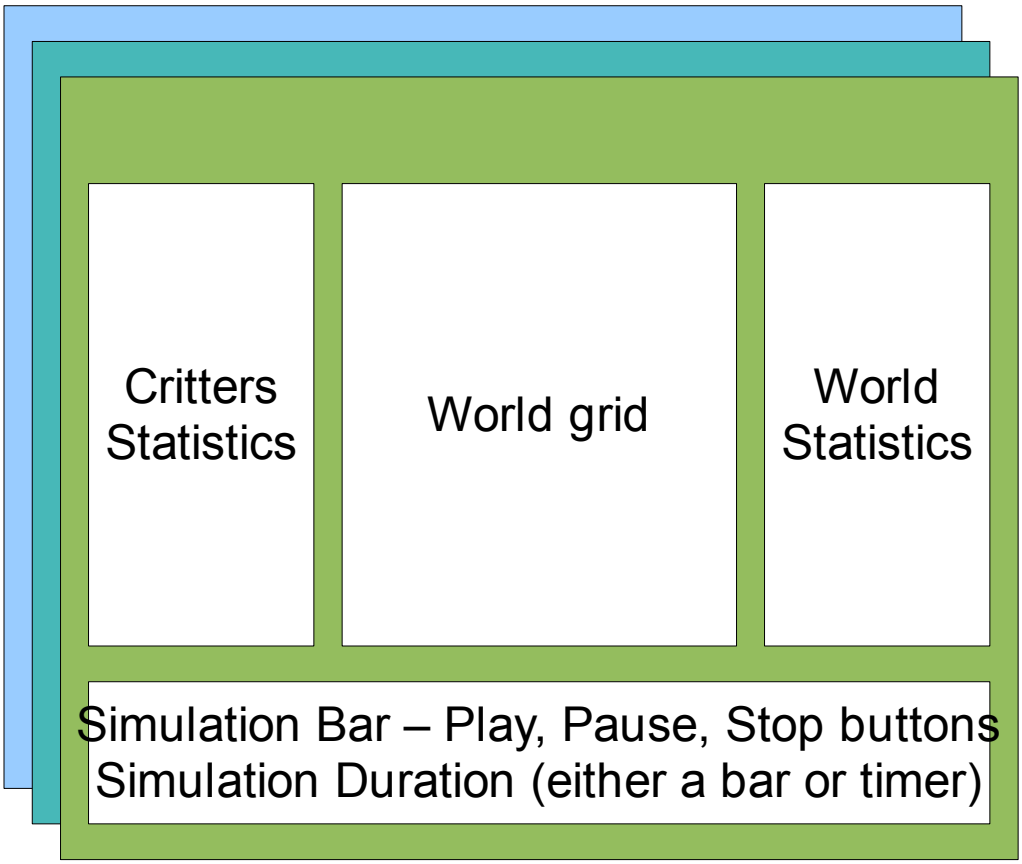
1.4.2Proposed Customization Screen (Wireframe)



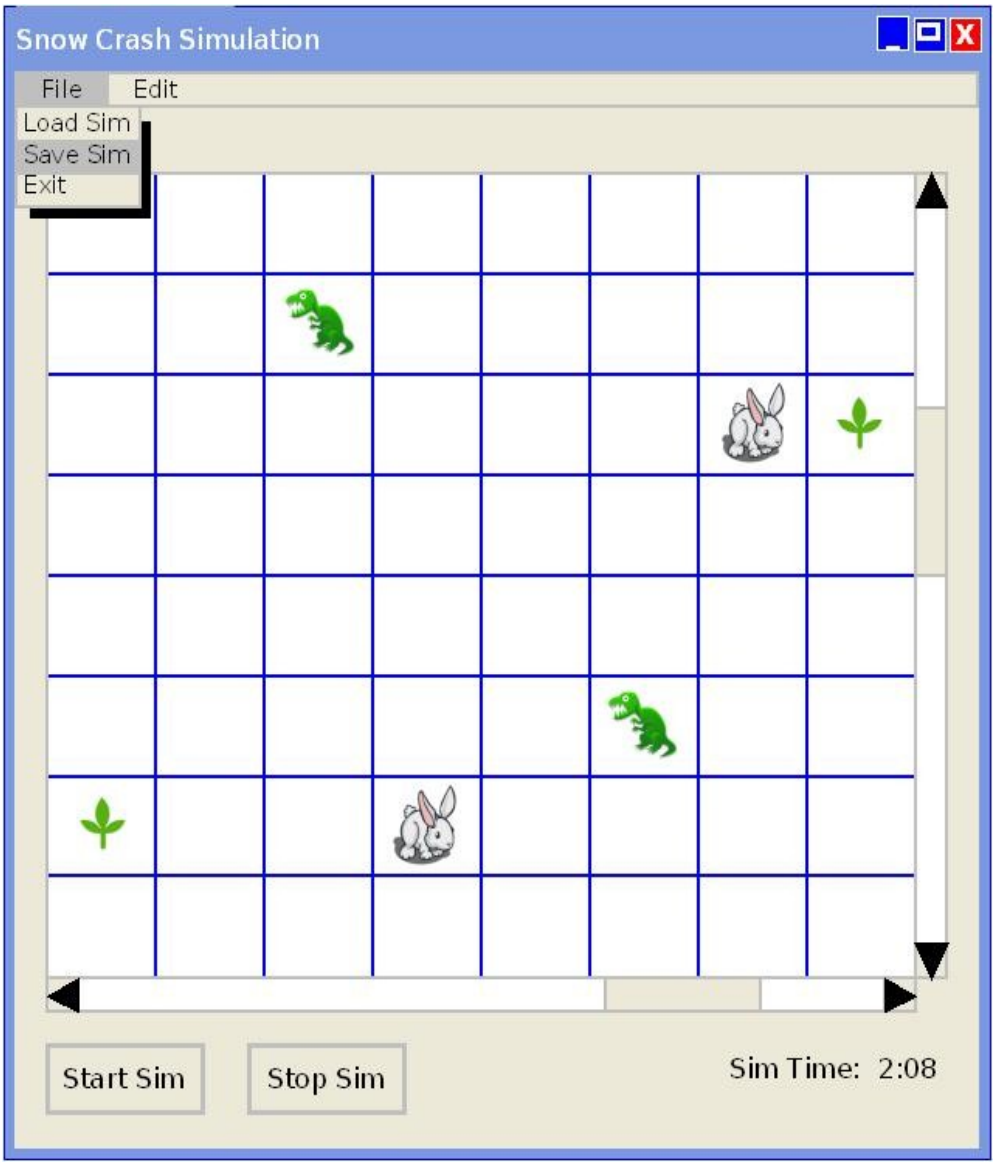
1.4.3Proposed Simulation Screen (Wireframe)



1.4.4Proposed Results Screen (Wireframe)



1.4.5UI Mockup(Simulation Screen)



1.5Basic Architecture

The application is a Java-based desktop application. The UI relies on a Swing interface and there are three primary screens: the "setup" screen (where critters are created and saved simulations are loaded into the UI), the "simulation" screen (where simulations are run), and the "results" screen (where the results of the simulation are published).

Data describing the simulation are kept in a configuration file loaded on startup. This configuration data is cached so that the simulation does not have to make IO requests every time it needs data. Simulations and critters are saved to disk as files as well.

The mechanics of Predators and Prey finding food are as follows.
To spot:

- Check vision – camouflage + random number;
- If check is > 0, spotted.

To catch (only for Predator and Prey):

- Check (PredatorVision – Prey Vision)/Predator Speed + (Predator Speed * distance)/ (Predator Speed – Prey Speed)
- If check < Predator Endurance, caught.

1.6Risks

There are a few risks involved in this project. They are the file IO, caching, the UI, and the world mechanics.

1.6.1File IO

The File IO is problematic as the file format has not yet been finalized. We need to model a more descriptive view of a trait, a critter, and a saved simulation to ensure we can write these models to disk and load. We have studied using CSV as our format but are concerned it is not robust enough. We'll also be looking into using XML, random access, and JSON.

1.6.2Caching

Related to the above, we need to model a cache to ensure that the cache is well organized as the system will be highly dependent on the cache to provide data.

1.6.3GUI

The GUI has been identified as a particularly risky piece of the application as there is not a lot of Swing UI experience in the group. To mitigate this risk we will be identifying the pieces of functionality we need and devoting one team member to prototyping and developing the UI components.

1.6.4Mechanics

Another risk identified is with the mechanics and how they adjudicate critter behavior. If the model and mechanics are wrong, then the critter behavior and world interactions will be unbelievable.

Also affecting the world mechanics are how we model the Critters to interact with each other on the world map.

2.Hours

2.1Total Hours

Deliverable	Estimated Hours	Actual Hours
STCD	30	25
SPD	40	38
SCMP	40	
SPMP	40	
SRS	60	
SDD	40	
Group Project Presentation	30	
Individual Project Report	10	

2.2Hours Broken Down

Item	Dale Earnest	Mike McWilliams	Jeff Dunn	Dong Luo	Total Hrs.
Meetings	2.5	2.5	2.5	2.5	10
Communications Plan	0	0	0	0	0
Application Investigation	2	3	4	4	13
Prototype (coding)	0	0	0	1	1
Prototype (drawings)	0	1	4	0	5
Project Proposal	4	0	0	0	4
Document Editing	1	0.5	0.5	0.5	2.5
Document Review	1	0.5	0.5	0.5	2.5
Total Hours	10.5	7.5	11.5	8.5	38