Fox and Rabbit Population Simulation Software



*Specification for a program to demonstrate the effects of rabbit- and fox-hunting policies on the populations of these animals for the Spring 2018 Capstone by*

Zachary Bolt and Benjamin Lewis

Table of Contents:

Executive summary……….……………………………………………………………………....3

System Function………...………………………………………………………………………....4

Benefits……………………………………………………………………….………..4

Target Client…………………………………………………………………………...4

User Roles……………………………………………………………………………..5

Use Case Diagram .......…………………………………………………………………………....6

Test Approach...…………………………………………………………………………………...7

Business Model Process Notation………………………………………………………………....8

Phases of Development …………………………………………………………………………...9

Glossary………………………………………………………………………………………….10

Works Cited…………………………………………………..………………………………….12

Executive Summary

* + **Our Mission**

Our mission is to create a desktop application that simulates the interactions between the populations of rabbits and foxes in an isolated community.

* + **The Company/Management**

The management of the company will be handled by the development team and the structure to be set as a two man collaboratory effort within creating the application. The workload will be distributed evenly and fall under each developer's preference to what work they are best suited for.

* + **Our Services**

Our services include one desktop application (app). This will include creating code for the app and to maintain all systems associated with it. This service includes maintaining the app.

* + **The Market**

The goal is to get this software in the hands of children in classrooms, and to have it used in natural museums, in order to educate the young and the interested about the fragile nature of our environment and necessity of hunting regulations.

* + **Our Competitive Advantages**

The currently available population simulation software is not at all kid friendly and quite expensive. This software is has an advantage in that simulations can be easily understood by even elementary-aged students.

* + **Financial Projections**

We do not aim to profit from this software. The app is to be freely distributed. The source code is also publicly available.

* + **Start-Up Financing Requirements**

The initial financial projection for the company is absolutely nothing to very low . We currently have access to all the resources needed through the university. But in a non-university setting the cost could depend on the equipment used by the company/development team such as computers, servers, office space, wages, and the website domain from which to distribute the software.

System Function

The main function of our app system is to allow for users to change the dynamic variables in legislating hunting, primarily tag limits, and see the effects on the populations of simulated groups of rabbits and foxes. The user will be able to change the starting characteristics for the fox, rabbit, and hunter groups to see how different ratios affect each group. The length of time that the user is able to maintain a balance before a catastrophe, such as rabbits or foxes being wiped out or repopulating to the point of being a nuisance, is the user’s “score.”

Gameplay can be broken down into “rounds.” Every round, the player is presented with a visual representation of the current populations. Then, the user is prompted to either increase or decrease tag limits, or do neither. After the user makes a decision, the app will take the data and apply our algorithm to update the internal values for the population numbers. If a catastrophe has occurred, then the simulation is over and the user is presented with their score. If a catastrophe has not occurred, then the next round begins. This continues indefinitely.

Our algorithm for updating the population values are as follows:

//Update Rabbit Variables

Rabbits.rCount = Rabbits.rCount \* Rabbits.rBreedSpeed;

Rabbits.rCount = Rabbits.rCount

- (Commissioner.rTagLimit \* Hunters.rHunterEffectiveness)

//(Hunters.rHunterCount \* Hunters.rHunterEffectiveness)

- (Foxes.fCount\*Foxes.fEffectiveness);

if (Rabbits.rCount < 0)

Rabbits.rCount = 0;

//Update Fox Variables

Foxes.fCount = Foxes.fCount

- (Commissioner.fTagLimit \* Hunters.fHunterEffectiveness);

Foxes.fCount = Foxes.fCount

+ (Foxes.fCount\*Foxes.fEffectiveness / Foxes.fRabbitsEatenToGrow);

if (Foxes.fCount < 0)

Foxes.fCount = 0;

**Benefits**

The main benefit of our project is to provide a fun and educational experience focusing on predator-prey relations and the impact of hunting.

**Target Client**

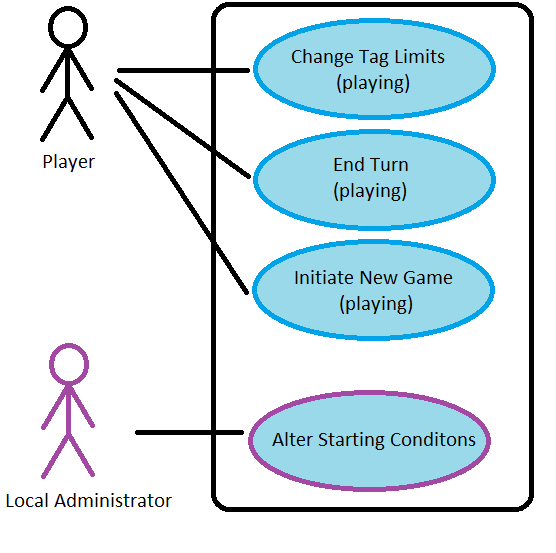
Our target client(s) is primarily elementary schools and nature centers, but also anyone that wants to have a fun and educational experience that enjoys management-style games or is interested in wildlife and hunting. We seek to target only a small portion of the educational gaming market, so our clients are limited to PC-based users. The client we are targeting is the elementary-school-aged child. That would mean more specifically, English speakers between the ages of 6 and 11. Our goal is to make the app approachable to the youngest of computer-users, so the art is highly representative and the game functions are simple. This is done by how our app is so simple that even kids can pick it up, while being challenging enough to not deter the older user base that is present in the mobile-game market.

**User Roles**

The user roles consist of only the players playing the game, and the local administrator. Players can only play the game, where the local administrator can make changes to the starting data file to customize the experience. As described in the system function section, the user will have the lowest level access possible to minimize risk of changing the starting file irreparably.

The administrative user roles will consists of the highest level functions in order to fix starting data. These functions will be only available to the administrators and can be used at their discretion to allow for fixing problems that may arise from the players of the game and from bugs within the game’s code. Overall, the administrative role is just a supervising user that will maintain the system and the game.

Use Case Diagram



**Player**: Refers to every individual using the software (playing the game). This is the typical user and has 3 basic interactions:

1. Changing Tag Limits
   1. Clicking a button in the app that alter variables in our algorithm
2. Ending the turn
   1. Clicking a button in the app that starts the next turn
3. Initiating a New Game
   1. Clicking a button in the app to reset the variables to the local starting data and start over
   2. Starting the application from the desktop

**Local Administrator**: Refers to a player that has elevated permissions. They can do anything a player can do, but can also change the application’s initialization data, which affects gameplay.

Test Approach

Is player input processed appropriately?

* When the “end turn” button is clicked, the turn counter should increase by one, and so should the “authority” score.
* When the user changes the tag limit for either foxes or rabbits by clicking the corresponding buttons, the displayed tag limit should change accordingly, and the user authority should decrease accordingly.
* The user should not be able to change tag limits when they have an inadequate amount of authority
* The rabbit and fox count at the end of each turn should represent the values after the algorithm has been applied.

Is the visual representation accurate?

* When the game starts, are the correct number of rabbit and foxe totems spawned?
* When the populations decrease, do the killed totems disappear?

Can the Local Administrator reliably change the simulation starting conditions?

* Does the number of totems being spawned at the start of the game accurate to the starting data file?
* Is there a starting data file for the Local Administrator to alter?
* Is the starting data file coherent enough for the Local Administrator to understand the changes that they are making?

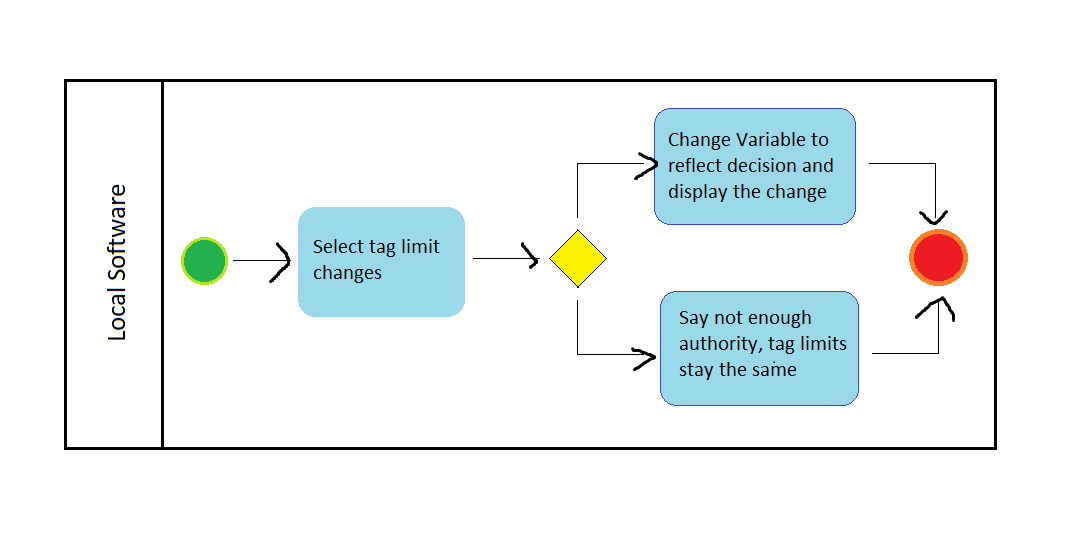
Are the sprites properly assigned to their designated object?

* This includes checking for every type of rabbit, fox, and UI sprites.
* The sprites must also be clearly unique and recognizable to the user so that there is no confusion regarding what object each sprite represents.

BPMN

### Business Process Model and Notation

**Playing the Game**



Phases of Development

Phase 1.

**MATH.** (2/20/18)

Mathematical understanding of the relationships we want to represent.

Which variables are involved?

Which are constant? Dynamic?

Phase 2.

**SKELETON.** (3/2/18)

Game functioning at data level, no text or visualization.

Implement the algorithms in a C# program running in unity.

Phase 3.

**MAKE PLAY.** (3/16/18)

Create a whole bunch of fun events and decisions for the user to choose from.

Create a canvas-based user interface.

Dynamic Tag Limit Values

Phase 4.

**MAKE FLESH.** (4/4/18)

Unity animations for the canvas-based user interface.

3D models of foxes, rabbits, and baskets

Spawning of foxes and rabbits in response to changes in populations.

Phase 4.5

**LEGITIMIZE.**

Package the executable in a way that is easily distributed and understood.

Auxiliary “installation” program.

Phase 5.

**MAKE TRUE.**

Add realistic data and the option to import custom starting data.

Glossary

C

The prefix “c” (unused) *refers to variables related to a relationship with the Commissioner class*

Commissioner - *The class that holds tag limit data*

Commissioner.authority - *Decision making credit*

Commissioner.aRegen - *the rate at which the player gains authority*

Commissioner.fTagLimit - *tag limit for foxes*

Commissioner.rTagLimit - *tag limit for rabbit*

E

Effectiveness - *see efficiency*

Efficiency - *see effectiveness*

F

The prefix “f” *refers to variables related to a relationship with the “FoxPop” class.*

Foxes - *Name given to class containing reproductive variables for foxes*

Foxes.fCount - *The current number of foxes*

Foxes.fEfficiency - *the effectiveness of a fox at killing rabbits*

Foxes.fRabbitsEatenToGrow - *the number of rabbits that must be killed by foxes in order for a new fox to be born*

FoxPop - *class containing reproductive variables for foxes itself*

Fox Totem - *visual representation of a fox or some fixed quantity of foxes*

H

Hunters - *the name given to the class containing hunter effectiveness data*

Hunters.rHunterCount - *the current number of rabbit hunters*

Hunters.rHunterEffectiveness - *factor in number of rabbits killed by hunters*

Hunters.fHunterCount - *the current number of fox hunters*

Hunters.fHunterEffectiveness - *factor in number of foxes killed by hunters*

HunterPop - *the class containing hunter effectiveness data itself*

R

The prefix “r” *refers to varriables related to a relationship with the “RabbitPop” class.*

Rabbits - *name given to class containing rabit reproductive variables*

Rabbits.rCount - *the current number of rabbits*

Rabbits.rBreedSpeed - *the baseline rate at which rabbits multiply*

RabbitPop - *class containing rabbit reproductive variables itself*

Rabbit Totem - *visual representation of a fixed number of rabbits*

RandomEventGen - *function that temporarily changes certain variables*

T

Turn - *cycles of applied reproductive algorithm*

Totem - *visual representation of either a rabbit or a fox*

Works Cited

15 things you (probably) didn't know about fox hunting. (n.d.). Retrieved March 13, 2018, from http://www.countryfile.com/explore-countryside/wildlife/foxhunting-facts

Cottontail Rabbit. (n.d.). Retrieved March 13, 2018, from https://www.in.gov/dnr/fishwild/3375.htm

Fox Hunting In America. (n.d.). Retrieved March 13, 2018, from https://www.americanheritage.com/content/fox-hunting-america

Rabbit Predators. (n.d.). Retrieved March 13, 2018, from http://www.rabbitmatters.com/rabbit-predators.html

Red Lynx. (n.d.). Retrieved April 13, 2018, from <http://scit.us/redlynx/>