Simulation Proposal

**Title: “Wolves vs. Sheep”**

**Project Goal**

*To design a program which will illustrate a simulation of the natural struggle between an apex predator, the wolf, and a flock of simple minded sheep guarded by a trusty hound.*

**Description**

*In this project, the simulation will generate a two-dimensional screen on which to display three types of “animal” objects. There will be the wolf, sheep, and the sheep dog, denoted by a square. All the shapes will have captions naming them “wolf”, “flock”, and “hound”.*

*In the simulations, the animal objects will have certain properties like “strength” and “hunger” which will determine their behaviors. The sheep dog will circle near the flock in hopes of deterring the wolf from its hunt. The wolf, attempting to stay well fed, will search for every opportunity to attack, kill, and eat the sheep. If a confrontation between the wolf and the dog ensues, the randomized property of strength will determine who wins the dispute. If the sheep populate to a point of saturation, the simulation will end in favor of the sheep. If the wolf manages to kill all the other objects, the simulation will end in favor of the wolf. If the dog manages to kill the wolf, the simulation will end in favor of the dog.*

*The need for this project is to produce content that is realistic and lifelike. The opportunity presented by this simulation is that of creativity. There will be endless areas for imaginative interactions between the wolves, the sheep, and the sheep dog using properties given to the objects. The main motivation for this project is to push creative boundaries. Goals for this project include: Exhausting every possible creative quality that can be given to the objects, demonstrating results that model real life accounts, and giving the animals realistic movement patterns.*

**Client/target audience**

*The target audience will be people of all ages and demographics. Seeing as the program will run free of human interaction, there is not much of an opportunity to attract gamers and other people seeking interface.*

**Team members and roles**

*Andrew Thomas, Analyst,* [Andrew.thomas0209@gmail.com](mailto:Andrew.thomas0209@gmail.com)

*Marisa Gilman, Project Manager,* [mgilman@terpmail.umd.edu](mailto:mgilman@terpmail.umd.edu)

*Colin Sullivan, Researcher,* [colinsully1@gmail.com](mailto:colinsully1@gmail.com)

*Jasmine Sullivan, Tester, jassyjasxoxo@aim.com*

* *Project manager – keeps track of individual task commitments, due dates and status (complete, in-progress, overdue, etc.). Reminds (and sometimes nudges) team members on their tasks. Reports weekly team update. Needs to be organized, a good communicator and willing to “push” people a bit.*
* *Analyst – Leads team members in analyzing the problem, breaking it down into sub-problems, identifying system components, creating a description and/or diagram of the components (modules, functions, etc.), and how they fit together. Makes sure that all the code produced by the individual coder fits together. This role especially often draws on the computational thinking strategies (although they all do to some extent).*
* *Tester – Determines whether the code meets the requirements. Identifies what parts of the requirements are satisfied, and what parts are not yet satisfied. Makes sure that new code doesn’t break the old functionality – that the system still works.*
* *Researcher – Gathers the data or information needed. For example, gets sample data from clients, finds Python functions or modules that do specific functions needed by the team, etc.*
* *Other roles – you may identify additional important roles*

**Required capabilities**

*A numbered list of tasks that users will be able to perform using the delivered system. For each capability, provide a description of the task. You may include sketches or screen mockups here. If you are familiar with scenarios or use cases, you may use either of those formats.*

*As stated above, the user will not in any way interact with the program. This means that the interface will not require much coding, if any at all. However, the objects will require a substantial amount of programming. Their functions are listed below.*

1. *Animal Motion: The animals must follow respective patterns of motion and should look fluid in the process of moving. Each animal will have its own method of motion. The sheep will generally stay together in a circular bounded flock labeled “flock”, but when the total sheep population becomes large enough, sheep that deviate from the masses will roam the screen. The wolf will freely move about the board in search of the flock and any rogue sheep. The sheep dog will operate around the circular bounds of the flock but with a larger radius.*
2. *Animal Interaction: To populate the sheep, upon collision with each other, there will be a certain percentage chance that a new sheep will be produced. This percentage will be adjusted to make the growth rate realistic. When the sheep dog and the wolf interact, there will be a randomized quality of “strength” applied to both the objects at the beginning of the game which will determine if the wolf or dog will die. If the wolf either slips by the dog, or manages to kill it, he will still be required to hunt and kill all sheep. Upon collision with the wolf, the sheep will disappear from the screen.*
3. *Hunger: The wolf will be given a set amount of hunger, starting at 0.0, which will slowly increase to 10.0 without a fresh kill. Upon reaching a hunger level of 10.0, the wolf will perish.*

**Additional capabilities**

*“Grid-like” movement patterns.*

**Needed materials and sources**

*Random movement code.*