Predator Prey Simulation[1]

Boe Zienko & Brenden Swope (Working with Dr. Becnel & Dr. Nick Long)
Stephen F. Austin State University

Abstract

This project was designed to effectively demonstrate how mathematics can be used to define a movement area for simulating a predator chasing its prey. Our goal is to create a simulation that closely resembles real-life scenarios. To achieve this, we have chosen to make both the predator and prey move simultaneously within this defined movement area. The project serves as a tool to provide a different perspective on the mathematics behind movement areas, helping individuals who may struggle to grasp these concepts through traditional equations.

Introduction

This Project is a Unity-based simulation that allows the user to control both a Predator and Prey in a turn-based system. We use Unity as our development tool to create our project with. Unity is a programming software that allows the user to create projects in 2D or 3D. Unity also allows for real-time manipulation of your project, such as a simulation. This lets the user move the different aspects, such as the models, around as needed. Unity, along with the previously mentioned tools, comes with a plethora of other tools available to all users to help them create projects.

When the project is launched, you will be greeted by the main menu screen. From there, you can start the simulation, change the settings in the options menu, or exit the game.

The options panel allows the user to control several aspects of the simulation, such as the number of prey and predators, the number of turns, and some of the properties that control the movement area.

Once the simulation is started, then, you can either zoom in, zoom out, or reset the camera. You can also control the avatars, starting with the prey avatars, and then you move the predators. Once both sides are selected, they move, and the turn counter increases by 1. There are 2 ways for the simulation to end: 1) the prey survives until the round counter runs out, at which time the Prey will win. 2) the predators can also take out all the prey, and then the predators win.

Figure. Predator/Prey Simulation Running

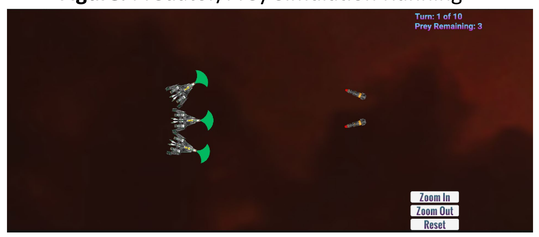
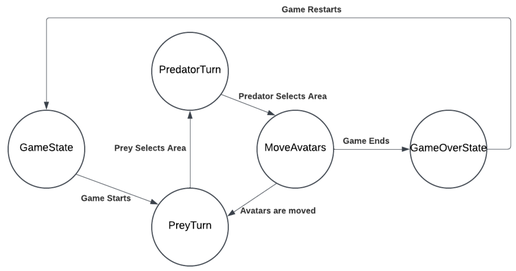


Figure. Game State Diagram



Governing Equations

The following equations are used to create the Legal Move area of an object based on the direction the object is facing and the parameters determining the maximum travel distance and turning radius.

Equation 1. Parameterization of the movement area mesh

$$(a,t) \;\; -> \;\; (\frac{1}{a}(1-\cos(dat), \frac{1}{a}\sin(dat))$$

- *a* represents the different angles of movement and ranges from $-1/R$ to $1/R$, where R is the max turning radius.
- *t* is where an object is in its progression along the path
- *d* is the max travel distance

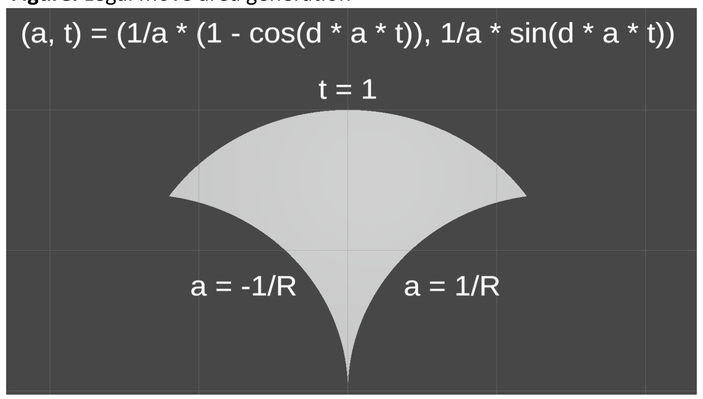
Equation 2. Inverse of x and y coordinate

$$a = \frac{2x}{x^2+y^2} \qquad t = \frac{(x^2+y^2)\arctan(\frac{x}{y})}{dx}$$

- *x* and *y* are the coordinates where the player clicks

Equation 2 are used to map the clicked point back to the path parameters, so the path can be created for the objects to follow.

Figure. Legal move area generation



State Design Patterns

The **State Design Pattern** is a behavioral design pattern used to allow an object to change its behavior when its internal state changes, as if the object has changed its class. It is especially useful when an object needs to change its behavior dynamically at runtime based on its state.
Here's a simple breakdown of how it works:
1.Context: This is the object that will change behavior based on its state. The context holds a reference to the current state.
2.State Interface: Defines the common interface for all possible states, listing the methods that all states must implement.
3.Concrete States: These are the actual classes that represent different states. Each concrete state class provides specific behavior for the methods defined in the state interface.
4.Transitioning: The context delegates behavior to the current state object, and each state can transition the context to a different state.

The base state is just where the avatars start and the other states are called GameState, GameOverState, PreyTurn, PredatorTurn, and MoveAvatars.

Simulation Options and Rules

Options:
-Number of Predators & Prey (1-10)
-Number of Turns (5-20)
-Turn Radius for Predators & Prey (0.5-5.0)
-Max Travel Distance for Predators & Prey (0.5-5.0)
-Proximity (Close, Mid, Far)
-Background
-Predator and Prey Type

Rules:
-Prey Starts first
-User must select an area inside of the move area (highlighted area)
-Both Predator and Prey must select their movement area before the simulation advances and moves both at the same time
-The Predator must connect with the Prey to capture it and win the game

Future Work

As of now our project is working in the 2D model, however we do not want to just stop there. We have plans for the future to later transform the project to not just a 3D model but also implementing it into virtual reality. This translation from a 2D model to a 3D VR model will allow the users to get a greater understanding of our simulation and hopefully see it from another light.

Contact

Boe Zienko / Brenden Swope
Stephen F. Austin State University
Computer Science
zienkoba@jacks.sfasu.edu / swopeb@jacks.sfasu.edu
(936) 205-7371/ (832) 926-2044

References

1. SFA-CS/PredatorPreyProject (github.com)