

Capture The Flag

Software Design Document

Keith Garfield

June 1, 2021

CS 225, Fall 2021

Embry-Riddle Aeronautical University

Daytona Beach campus

1 Aerospace Boulevard

Daytona Beach, FL 32114

INTRODUCTION:

This Software Design Document (SDD) describes the functionality and software design of a Capture The Flag (CTF) game. In a capture the flag game, two teams compete to attempt to capture the opponent's flag and carry it to a home base. The CTF developed for this project allows for a human user to select a team of players, which then compete against a computer generated team. The players are autonomous agents, meaning that the human player has no direct control over them once the game begins. The human creates a team by selecting from agents of varying abilities and behaviors provided within the game. Note: The term "player" will be used to refer to the human player, and the term "agent" will be used to refer to the autonomous agents participating in the CTF.

This CTF project was created primarily as an example project in Embry-Riddle Aeronautical University's (ERAU) Computer Science II course (CS 225). This project contains all features expected of a student project, to include GUI interaction, file input and output, exception handling, and the use of inheritance.

This project was also used as a programming exercise in summer Coding Camps offered by ERAU.

This document contains two main section: a Problem Description section describing the rules of the CTF game as implemented, and a Problem Solution section presenting a description of the CTF software.

PROBLEM DESCRIPTION:

This project implements a CTF game in which teams of autonomous agents compete against each other. The game is played in a rectangular, two dimensional space with no obstacles or terrain features, as shown in Figure 1. Each team has a home base, and a flag located at their home base at the start of the game. A team wins a single game by carrying the opponent's flag from the opponent home base to the team's own home base. Play continues until one team has won a match by winning three games.

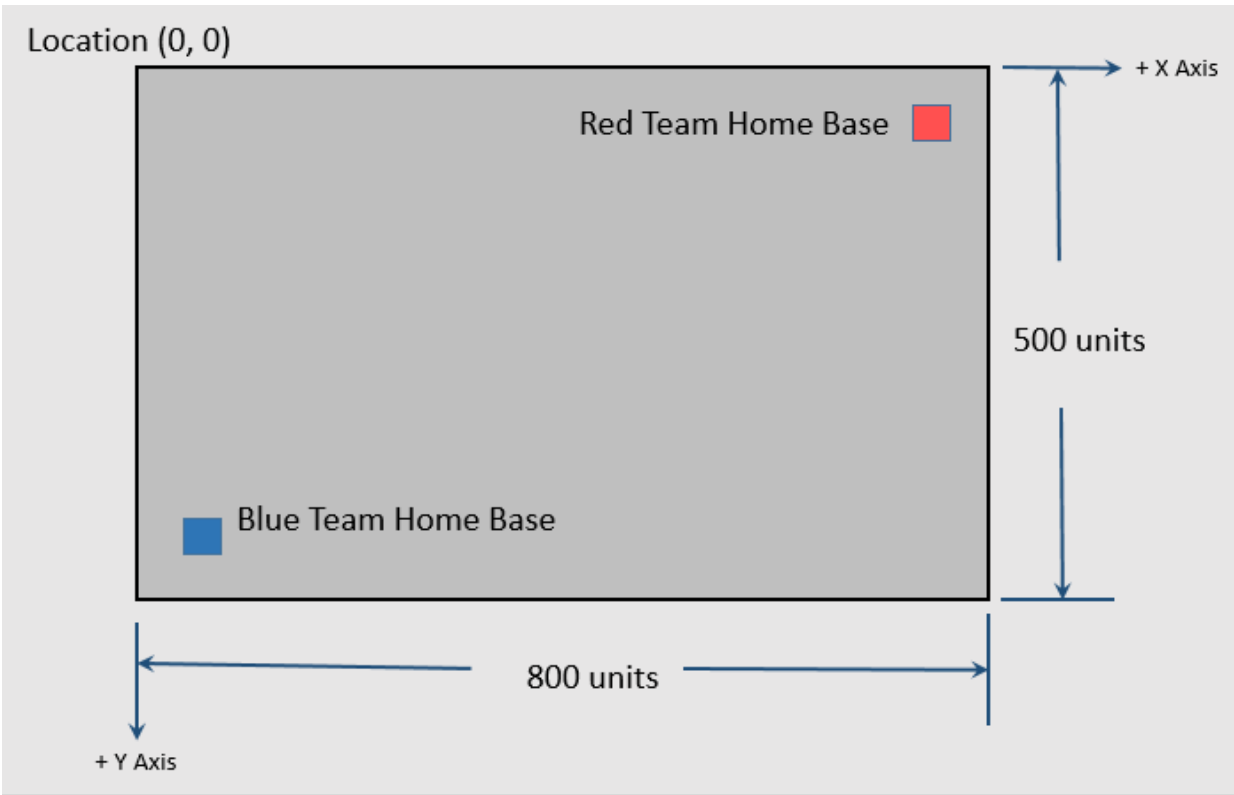


Figure 1: CTF Playing Field

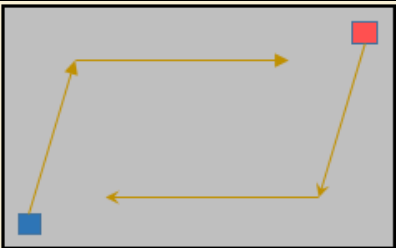
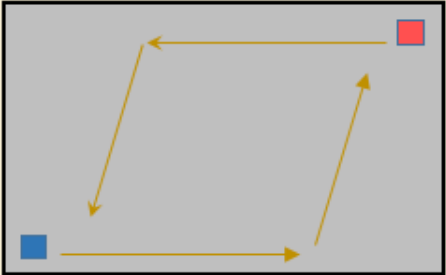
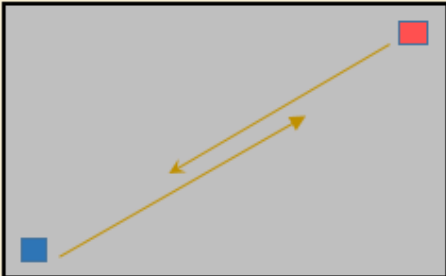
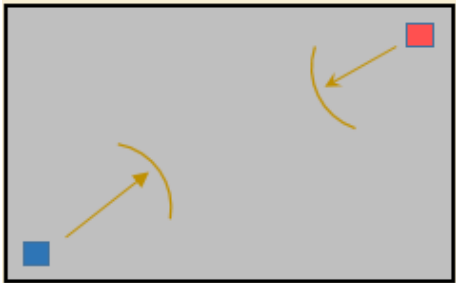
The player creates a team of agents by selecting from a pool of available agents of varying types. There are three types of agents: runners, hunters, and sleepers. Each agent has speed and strength attributes, and exhibit different capabilities and behaviors. Runners are able to capture the opponent's flag and carry back to the home base. Hunters are able to remove opposing agents from the game, but cannot carry a flag. Sleepers are able to cause opposing agents to "sleep" for a period of time, but cannot carry a flag. These capabilities are summarize in Table 1.

Table 1: Summary of Agent Types

Agent Type	Can Carry Flag?	Behavior After Completing Route
Runner	Yes	Always moves toward the opponents flag and attempts to carry it to home base.
Hunter	No	Moves toward closest opposing agent and attempt to remove it from the game by reducing strength attribute.
Sleeper	No	Moves toward closest opposing agent and causes it to become inactive (sleep) for a number of ticks.

The player selects how many of each type of agent is on the team. In addition to selecting the agents, the player is able to modify the speed and strength attributes of each individual agent. The player also assigns a route to each agent. Routes are summarized in Table 2. Lastly, the player is able to delay the entry of each agent by assigning a time delay (in seconds) to each agent on the team. Players are able to save teams to a file for later use.

Table 2: CTF Routes

Route Name	Description	Visualization
Up and Out	Agent moves at an angle between 55 – 75 degrees for 250 – 350 units, then horizontally for 200 – 400 units.	
Out and Up	Agent moves horizontally for 200 – 400 units, then at an angle of 55 – 75 degrees for 250 – 350 units.	
Grand Central	Agent moves at an angle between 35 – 55 degrees for 400 – 550 units.	
Loiter	Agent moves at an angle of 15 – 75 degrees for 100 – 150 units.	
Notes: 1. Angle and unit values determined randomly within the ranges in the descriptions.		

The rules of the game are provided below (Note: Specific numerical values may be adjusted for game play once play testing begins.):

1. CTF is played on a rectangular field having a width = 800 units and height 500 units.
2. There are two teams: Red and Blue.
3. The Blue home base is at $(x, y) = (50, 450)$. See Figure 1.
4. The Red home base is at $(x, y) = (750, 50)$. See Figure 1.
5. The game is played in turns, referred to as "ticks."
6. An agent is an individual actor in the game.
7. There are three types of agents: runners, hunters, and sleepers.
8. A team is composed of seven agents.
9. The player selects how many of each type of agent is on the player's team.
10. There is no restriction on the number of each type of agent selected by the player, though the total must equal seven.
11. Each agent is assigned a speed and strength attribute value, expressed in integer values.
12. Speed and strength attribute values cannot sum to more than ten at the beginning of the game ($\text{speed} + \text{strength} = 10$).
13. An agent must be assigned a minimum speed of 1.
14. Agent speeds are fixed once game play begins.
15. All active agents (see Table 3) move the number of units designated by their speed in every tick.
16. An agent must be assigned a minimum strength of 1.
17. Agent strength may be reduced through game events (see Table 3).
18. An agent is removed from the game if its strength drops to zero or less.
19. Only runner agents may carry flags.
20. Runners "pick up" a flag by passing over it.
21. Runners may only carry the flag of the opposing team
22. Runners that are carrying a flag and are removed from the game will drop the flag at their location when removed from the game.
23. The player assigns a route to each agent on the team (see Table 3).
24. Agents will initially move according to their route, and then move and interact per the behaviors listed in Table 2.
25. The player assigns a time delay, in whole seconds, for each agent to enter the game.
26. All agents enter the game at their home base.
27. Agents with a time delay of zero begin the game in play.
28. Agents with a non-zero time delay will enter the game when the game clock is equal to their time delay.

29. Agent location refers to the (x, y) coordinates of the center point of the agent.
30. When two agents of opposing teams are located within 15 units of each other, they will interact per Table 3.
31. If multiple agents of an opposing team are within 15 units of an agent, the agent will only interact with the closest one.
32. A team wins a game when one of its agents carries the opposing teams flag within 15 units of its home base.
33. A match is won when a team wins three games.

Table 3: Agent Interactions

	Runner	Hunter	Sleeper
Runner	No effect	Both agents strength is reduced by the smaller of the two strength values.	The runner is “put to sleep” for a number of seconds equal to the strength of the sleeper. No change to strength attributes.
Hunter		Both agents strength is reduced by the smaller of the two strength values.	The sleeper agents strength is reduced by the strength of the hunter AND the hunter is “put to sleep” for a number of seconds equal to the sleeper strength prior to being reduced.
Sleeper			No effect.

PROBLEM SOLUTION:

Instructions: This section explains the software design. Provide an overview of the approach you took to the solution. Describe each class, the job that the class performs, and how it relates to the other classes. You may want to, or need to, provide algorithms for any complex computations. This discussion must refer to the Unified Modeling Language (UML) class diagram discussed below.

In addition to the descriptive text, you will supply a UML class diagram of your software design. The UML diagram is a simple pictorial representation of your design. The UML you provide must match the software that you turn in (it's normal to change the design a little as you go – don't forget to update the UML!). The UML must be embedded in this document – UML's submitted separately from this document receive a score of zero.

[Shall be delivered in near-complete draft version for P3, completed by deliverable P4, and edited as needed for future deliveries.]

REFERENCES:

All sources shall be cited in this section. If the project required no sources, keep this section but leave it blank. Sources might be papers and texts in the general problem domain of the project, code snippets, libraries incorporated in the project, or even algorithmic solutions to specific parts of the project.

[Shall be non-empty in P2 and updated as needed with each deliverable.]

APPENDICES:

This is optional, but may include external sources, source code, or other related material.

[Shall be completed as needed with each deliverable.]