AI System RPG

Jason Lao, Daniel Sabanov, Devin Toms, Kurt Grzybowski
CS 4624: Multimedia, Hypertext, and Information Access
Professor Edward A. Fox
Virginia Tech, Blacksburg VA 24061
May 03, 2023

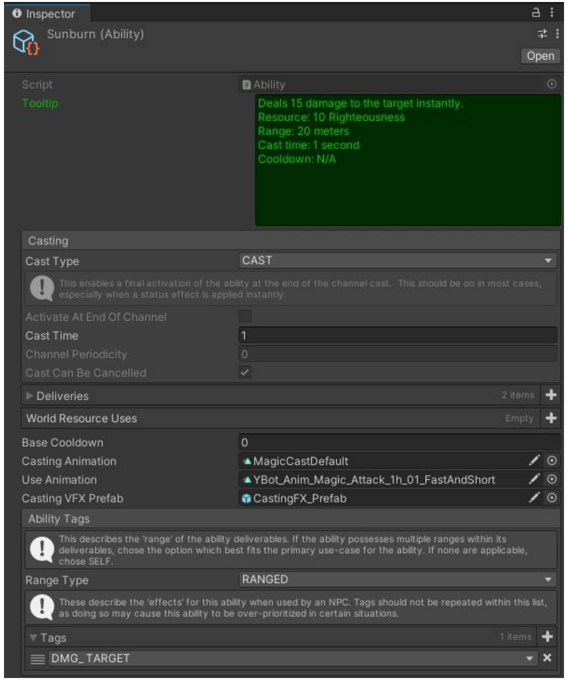
Outline

- Project Description
- Design
- Implementation
    - Routine System
    - Black Box
- Testing
- Future Work
- Lessons Learned
- Acknowledgements

A screenshot of the current build of the game.

Project Description

- We were tasked with creating an AI with the following capabilities:
  - The AI must be able to interact with the implemented ability system effectively
  - The AI is under the same restraints as a player character
  - The AI can either be an enemy or an ally
- We chose to use a routine-based solution, where we created a Black Box function to deduce the best course of action
- Ability Tags were created in order to better categorize the possible abilities the AI is capable of using and what they can do



Background

- Client: **Prof. Zach Duer**
  - Assistant prof. at VT School of Visual Arts
- Developing RPG for >1 year
  - Completed:
    - Modular ability system
    - Animation system
    - Multiplayer networking

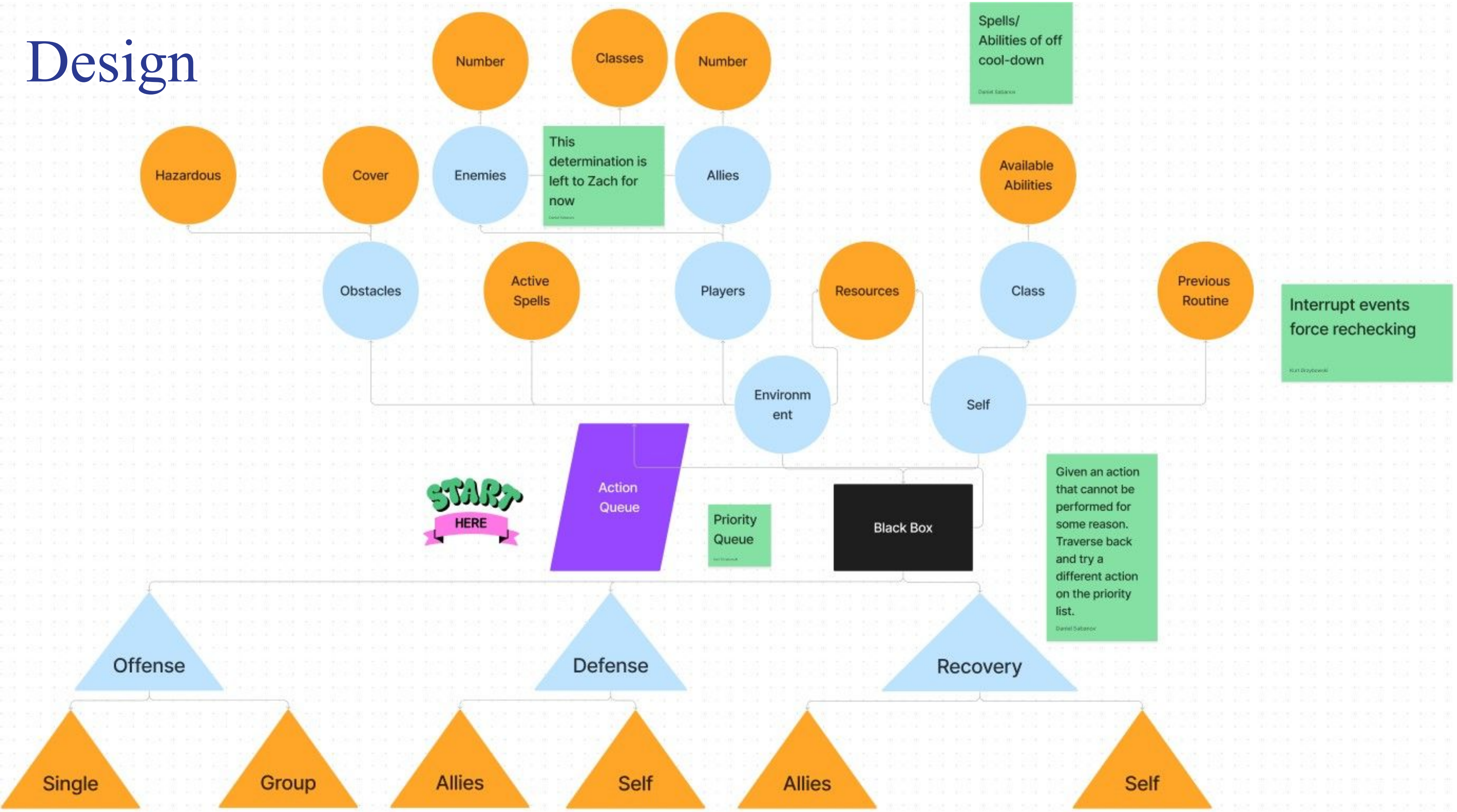
nothing_nomeanings

Square Dance Cubed

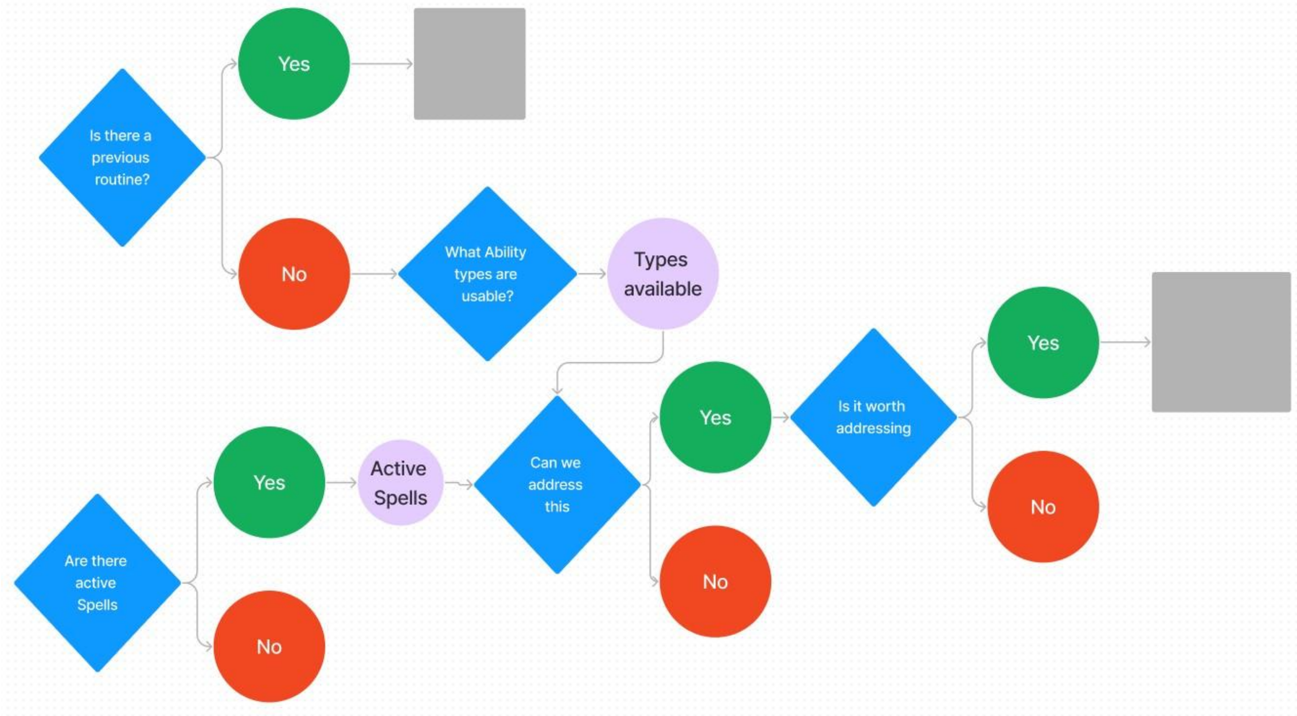
Requirements

- **NPC's need AI**
  - Engage with players and against players
- **Deal with Modular ability system**
  - No set abilities
- **Feel "Real"**
  - Same access to information as characters
  - Flexibly use abilities

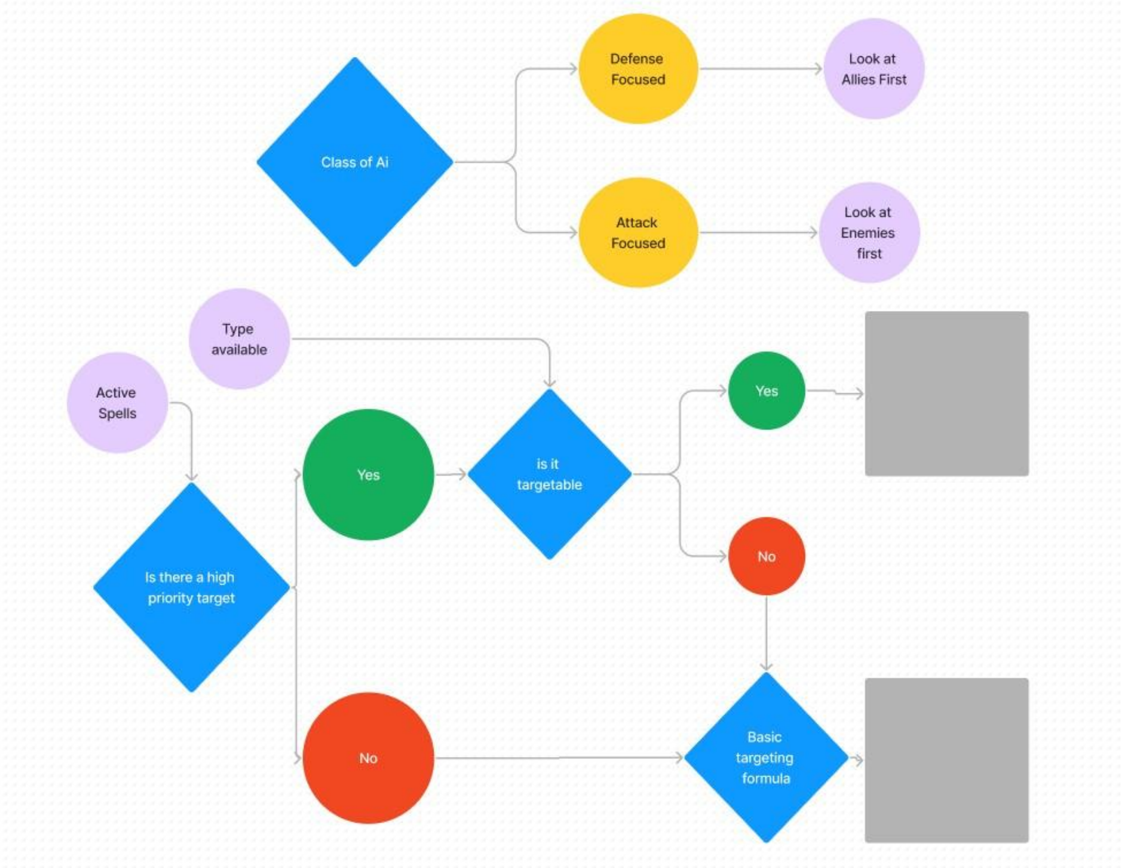
Design


Spells/ Abilities of off cool-down
Number
Classes
Number
Hazardous
Cover
Enemies
This determination is left to Zach for now
Allies
Available Abilities
Obstacles
Active Spells
Players
Resources
Class
Previous Routine
Interrupt events force rechecking
Environm ent
Self
START HERE
Action Queue
Priority Queue
Black Box
Given an action that cannot be performed for some reason. Traverse back and try a different action on the priority list.
Offense
Defense
Recovery
Single
Group
Allies
Self
Allies
Self

BlackBox


Is there a previous routine?
Yes
No
What Ability types are usable?
Types available
Can we address this
Yes
No
Is it worth addressing
Yes
No
Are there active Spells
Yes
No
Active Spells

BlackBox

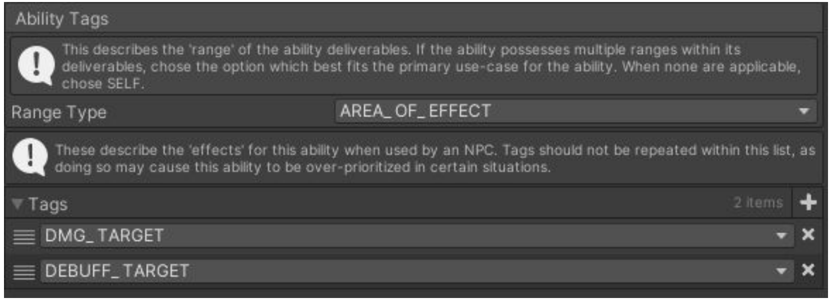

Class of Ai
Defense Focused
Look at Allies First
Attack Focused
Look at Enemies first
Type available
Active Spells
Yes
is it targetable
Yes
No
Is there a high priority target
No
Basic targeting formula

Implementation

- Four main things that were implemented:
    1. **Routines** - Controls AI behavior
    2. **Ability Queue** - A queue meant to hold multiple abilities to be used in succession (can only hold 3 abilities)
    3. **Ability Tags** - Categorization of abilities an AI is capable of using in order for the AI to know what it can do
    4. **Black Box** - A file that controls total AI behavior using all previously mentioned concepts, it is the AI’s “brain” so to speak

Routine System

- AI behavior controlled through a series of “states”
- `PickRoutine` state reads from Black Box to find best routine state
- Routine fills Ability Queue based on ability tags
- Ability tags customizable by designer




```
protected void BuildAbilityQueue()
{
    // get list of all currently useable abilities
    List<Ability> abilities = entityController.abilityList.GetAvailableAbilities();

    // refine list of useable abilities to match current routine
    List<Ability> validAbilities = GetRoutineAbilities(abilities);

    for (int i = 0; i < 3; i++)
    {
        entityController.pendingAbilities.Enqueue(ChooseAbility(validAbilities));
    }

    Debug.Log("Refilled Queue");
}
```

BlackBox

- Evaluates the environment and selects the best general strategy (**state**)
  - Attack
  - Defend
  - Heal

```
Frequently called   7 usages   hebrew
public RoutineInformation(State.ROUTINE nextState, EntityController target,
    RANGE range, URGENCY urgency, QUANTITY quantity)
{
    NextState = nextState;
    Target = target;
    Range = range;
    Urgency = urgency;
    Quantity = quantity;
}
```

```
public struct RoutineInformation
{
    9 usages   hebrew   3 exposing APIs   More...
    public enum RANGE
    {
        CLOSE,
        RANGED
    }

    9 usages   hebrew   4 exposing APIs
    public enum URGENCY
    {
        CONSERVATIVE,
        NORMAL,
        URGENT
    }

    9 usages   hebrew   4 exposing APIs
    public enum QUANTITY
    {
        GROUP,
        SINGLE,
        NONE
    }
```

BlackBox Logic

- Each time the character needs to decide on the next action, the `NextRoutine()` function is called.
- The function runs through a set of conditionals to evaluate the situation
- As soon as a condition is met, creates a `RoutineInformation` struct and returns
- These conditionals can be adjusted by interacting with prespecified global parameters

```
private const float HealthThreshold = 0.25f;
private const float GroupingThreshold = 3;
private const float PrimaryTargetPriority = 2;
private const float SingleMemberImportance = 0.2f;
private const float Agressivenesss = 0;
```

Testing

- For our testing, Professor Duer provided us with various scripts in which we can build and create various scenarios for testing purposes. This provides us with a great deal of control over testing for the Black Box and the Routine System:
    - We can control how NPCs are spawned
    - We can control where NPCs appear
    - We can control what kind of NPCs are made, enemies or allies
- This allows us to test AOE attacks, healing allies, crowd control abilities, and test attack range, damage, radius, etc.
- The game's user interface (UI) elements and Unity's built-in console provide another potential testing tool, and can be used to test AI instances interactions on a more fine-grained level.

Future Work

```
protected Ability ChooseAbility(List<Ability> abilities)
{

    // TODO: any ability prioritization should go here
    // choose a random ability from the fully filtered list
    Ability chosenAbility = abilities[Random.Range(0, abilities.Count)];

    return chosenAbility;
}
```

- Next Few Weeks
  - Tweak any misfitting variables
  - Document non-implemented planned features
- Future Teams
  - Incorporate AI into both the Black Box and Routine System
  - Incorporate more variables from the environment
  - Add focus on non-combat situations

Acknowledgements

We would like to thank our client, Zach Duer, Dr. Fox, and the TAs for providing us with the opportunity to work on this project and providing us with the necessary resources to succeed.

References:

- https://sova.vt.edu/faculty/zduer/
- https://www.zachduer.com/