



Crushed

Game Artificial Intelligence CS5150

04.17.2018

Harshit Gupta

gupta.hars@husky.neu.edu

Northeastern University
360 Huntington Avenue
Boston, MA 02115

Wanhui Han

wan.ha@husky.neu.edu

Northeastern University
360 Huntington Avenue
Boston, MA 02115

Overview

Crushed is a RPG/Simulation game where a boy is trying to impress his crush. The player will play the role of the boy and collect objects to win the heart of the girl while fighting against the ex-boyfriend. The game will implement AI techniques such as A*, Goal-Orientation Behavior and Optimized Greedy Search.

Repository Link: <https://github.com/n33t1/Crushed.git>



Setup

To View Source Code:

1. You need to install Unity.
2. Download this repository.
3. Open this repository from Unity if you want to view the source code.

To Play:

1. cd Crushed\Build
2. Select Mac.app for MacOS
3. Select Windows\Crushed.exe for WindowsOS

Attributes

Mood Bar: There are three mood bars in total: friendship, happiness and romance. Your goal is trying to increase the two bars such that you can win the girl's heart. But be careful you don't want to be ended up as friend with the girl, so keep an eye on the friendship bar. Also, the girl's ex is trying to decrease the happiness and romance bars. Watch out for him!

Health Bar: You can shoot the girl's ex to freeze him for 30 seconds. But the guy will become desperate when the girl almost falls in love with you (meaning that he will chase you and try to terminate you). If your health bar goes to 0, you will lose.

Instructions

Keep in mind, your goal is to make Happiness and Mood Bar reach 100% and win the girl's heart (as well as the game).

1. Use arrow keys to navigate the player
2. Press Q when you reach an object to pick it up
3. Press E when you reach the girl to give her the object
4. Some objects can form combo when you give them one after the other. Cheat-sheet for the combos are available below.
5. Some objects raise only one bar while other objects raise both bars.
6. Some objects can reduce both bars. However, these objects can raise bar tremendously when used as combo.
7. The Girl's ex-boyfriend will also try to collect objects and give them to the girl to reduce her mood.
8. You can shoot the ex (using Space) to reduce his health. When his health goes to 0, it pauses for 30 seconds which gives you time to perform high value combos.
9. When both Happiness and Friendship bars are 100%, go to the girl and press P to propose and win.
10. Giving same object again and again will raise the friendship bar. If friendship bar goes to full, you get Friend zoned.

Team

[Harshit Gupta](#): level design, storytelling, goal-oriented behavior using sum of squares, optimized greedy search

[Amber Han](#): character design, implementation for pathfinding using navigation mesh, general implementation, player combat

Game Mechanics

Lose Conditions:

1. Both the happiness and romance bar go to 0%.
2. Friendship bar goes to 100% and you get FRIENDZONED.
3. Health Bar goes to 0%.

Win Conditions:

1. When both Happiness and Friendship bars are 100%, go to the girl and press P to propose and win.


Combos:

1. Red rose → Chocolate (1.5x Romance increase)
2. Red rose → Ring (1.5x Romance increase)
3. Chocolate → Ring (2x Romance increase)
4. Red rose → Chocolate → Ring (2x Happiness increase, 3x Romance increase)
5. Blue rose → Dress (1.5x Happiness increase)

AI Systems

Pathfinding with NavMesh/A*: We linked the unity pathfinding library to our game which implements A* such that our NPC can reach the destinations we assigned to him without any collision.

Object analysis: The NPC analyzes the mood change after a certain object is given to the girl and add the object to the correct list. If it is a new object, then the NPC will add it to the unexplored list. Otherwise, if the object helps the NPC reduce the mood bars, the object will be allocated to the correct mood list. Or if the object increases the mood bar meaning that it helps the player, then the object is added to the unsafe list and the object will not be traversed in the future.



Combo detection: We implemented a graph creation algorithm for the NPC. Whenever the mood reduces greater than the specified amount for that object, the NPC analyzed the current and previous objects and allocates the objects into the appropriate graph and position within the graph. For this, we implemented a graph for each mood, namely romantic and happiness. If a combo is for a particular mood, then the combo is added to the corresponding graph. If the combo is for both moods, it is added to both graphs.

Goal-oriented behavior for the combos: There are 4 behaviors in total: winning, safe, danger and critical. The execution of each behavior is determined from the sum of squares of all the mood values. The winning behavior prioritizes the exploration behavior with the combination of combo execution behavior. The safe behavior only focusses on exploration. The danger behavior prioritizes combo execution with the combination of exploration. The critical behavior only focusses on the chasing and shooting at the player.

Optimized greedy search: During the execution of the combos, the NPC keeps track of the highest optimal combo relative to each mood. During the danger behavior, the NPC compares the current optimal combo with the highest value combo obtained by traversing through the graph and obtaining the sum of individual combos. We have implemented the greedy search algorithm that focuses on optimizing the graph search by shortening the combo length and removing the redundant paths.

Behavior Diversion: When the player is close to winning the game, the NPC will enter the behavior of chasing the player and trying to shoot the player. The movement speed for the NPC will gradually increase or decrease based on the mood bars. The behavior of the NPC is completely diverted from trying to execute the combos and shifts to going on the offensive.

Failures and Setbacks

1. We tried to implement A* for pathfinding directly. But since the 2D map in Unity was not a graph, we swapped to NavMesh instead.
2. Our initial evaluation function for goal-oriented behavior was taking the original value and comparing it to the threshold. This prevented our NPC from differentiating between the emotions that need immediate attention.
3. We faced animation problems for player movement when the player moves diagonally. We fixed this problem by restricting the player movement in 4 directions, which eased the implementation of shooting.

4. We tweaked parameters for the player movement speed and shooting speed as well as NPC movement speed and wait time. We also tweaked the parameter for threshold values in goal-oriented behavior.
5. We also changed the collision radius to make the collision smooth between the NPC and the game objects.
6. We struggled with the correct data structure to use for our implementation of graph. We tried hash table, but it didn't allow to store multiple values for a given key. Eventually we used a dictionary with combination of list to implement the graph.

Successes

1. Our NPC is hard to defeat. The combo detection mechanic optimizes the damage he can do to the player. When the player gets closer to winning, it will increase the moving speed and doing more damage to the player. The behavior diversion allows him to forbid the player from winning by shooting at the player.
2. Our NPC can automatically detect the combos. We don't need to hardcode any combo. It can create a graph when it detects a combo and keeps track of the optimized combo to execute based on the current mood requirement.
3. Our NPC can accurately reach the objects with shortest path while avoiding collisions. The animation of our NPC and player regarding movement is smooth.
4. Even though all the objects are in 2D, we implemented 3D features by tweaking the collision and Z-axis values relative to the camera.

Evaluation

We had a play testing night with 10 volunteers. We received mostly positive feedbacks on the game experience with some bugs on NPC movement reported. Some of the comments are below.

Positive Feedbacks:

- "I really liked the concept. It is simple and super easy to learn the general idea, and it allows the player to find more combos."

- "The AI was easy and complex at the same time. As I proceeded, the enemy started moving faster and started shooting which made it difficult to win towards the end."
- "The AI killed me with bullet when I was about to win. It is a challenging game."

Negative Feedbacks:

- "The enemy takes a lot of time to get great combos. It executes good combos towards the end when the game is almost over."
- "There are some movement bugs for the AI. Like it gets stuck when there is very less space to move between player and girl."
- "Kind of hard to find the combos as the AI decrease the bars at the same time. You should give some tips."

Lessons Learned

- While we were trying to implement the adaptive learning, we realized the optimized greedy search is more relevant to our game than machine learning approach. We realized we can achieve similar effects with simpler approach. Complex algorithms do not mean high entertainments.
- It is easier to separate different AI mechanics in different scripts or classes rather than keeping them in the same script.
- Testing with game development is difficult in that many AI features are dependent on each other that makes unit testing complex. For example, we need to have pathfinding implemented first before we can test combo execution.