Grand Canyon University

Project 2 - Least Constraining Value Heuristic

Justin Dietrich and Ryan Scott

CST-415: AI in Games and Simulations Lecture & Lab

Dr. Ricardo Citro

January 30, 2022

A brief description of the game or simulation (one paragraph)

1. Our game will be from a top-down view, set in an environment that has paths, walls, and other types of cover or obstacles. So far, we know we will have the player move through some randomly generated grid-like levels, trying to find or complete the objective(s) while fighting off enemies that are placed throughout the level.

How are the concepts listed above relevant and its purpose? (one paragraph)

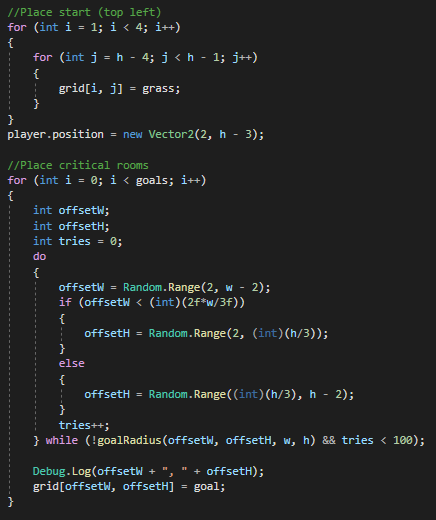
1. The game uses the concept of ‘least constraining value heuristic’ in the level generation. There are certain design choices in game levels that are important to consider, such as making it unique and non-linear, and most importantly possible. It’s hard to make sure levels meet these requirements when you don’t hard code each one, which is why it uses a tuned generation algorithm. The algorithm will place all of the required and most finicky objects on the map before generating the layout. With all of the more required things out of the way, there are a lot fewer possible map layouts for the program to guess and choose from, eliminating many possible failures.

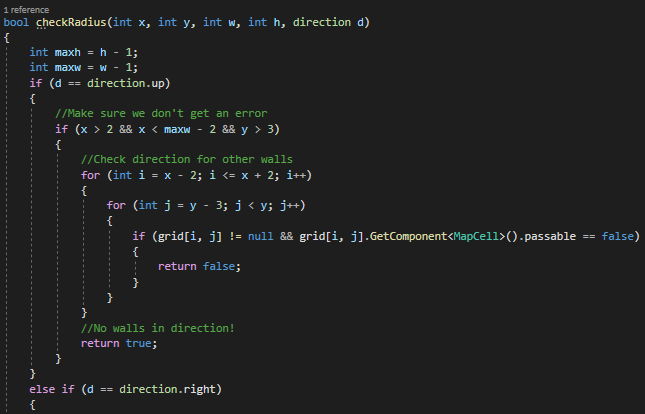
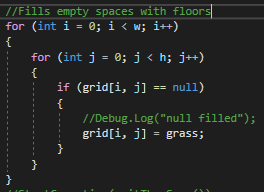
Which search method will be used? (one paragraph and bullet points outline)

1. The algorithm will most likely be cutset conditioning. The variables connected to the most other variables are done first. The variables chosen first will be the required rooms or objects, while the other variables are wall structures and patterns. The algorithm goes through the grid and sometimes checks to see if it can place a wall. If the tile is already occupied by an initially generated variable, it uses another possible solution for the tile and keeps going.

Github: <https://github.com/AsePlayer/CST-415>

Example Scripts

  
Critical rooms are placed first, since one of the walls’ constraints is that it cannot intersect a critical structure. A constraint of the goals is that they cannot be within 10 units of each other. This error is prevented by the goalRadius function;

1.   
   This is a part of the code that makes sure that there are no future errors. It looks to see if a wall is about to lock with another wall by looking at a few tiles ahead and checking if there is another wall in any.  
     
   Floors are inserted into every open space after all walls and rooms are generated, since there are no constraints as to where the floors can be.

How will you overcome unforeseen obstacles during implementation? What is your 'plan B'?

1. If we can’t get a good algorithm for the level generation going, we could always change it to where it’s either easier to code but less optimized/slower or with more linear and less unique level generation.

How is the project aligned with the current topic objectives?

1. The project shows that we can create a program that can guide itself into finding a solution for a task that we give it. For us, we task the program with making a level that has a randomly generated layout, with the constraint being that it is possible. That constraint is pretty broad, in that there are many things that go into making a level possible such as making sure the objectives aren’t blocked off by walls.

| Constrained variables | (Broad) Level has to somewhat resemble a labyrinth  Walls cannot block off parts of level  Walls cannot intersect critical structures  Goals cannot be too close to start  Goals cannot be too close to each other |
| --- | --- |
| Program finds solution | Map that meets all constraints is created with an algorithm that checks for future failure |
| Least Constraining Value | Floors are not constrained, so tiles are generated until floors are the only solution for every tile left. |

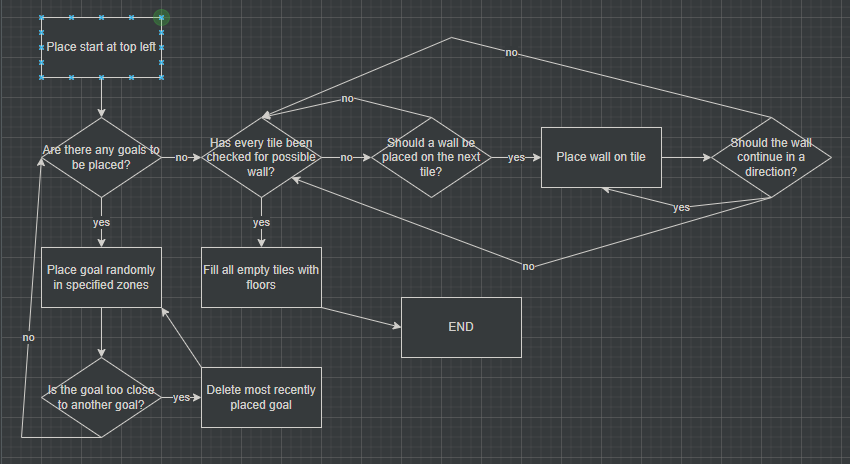
What will appear on the screen: animation, user interactions, information dashboards, UI elements, etc.

1. The maps are viewed from a top-down perspective, and the layout will appear gridlike. The player will be able to move through the level with expected results such as not being able to move through walls. Enemies will pathfind throughout the generated map to get to the player.

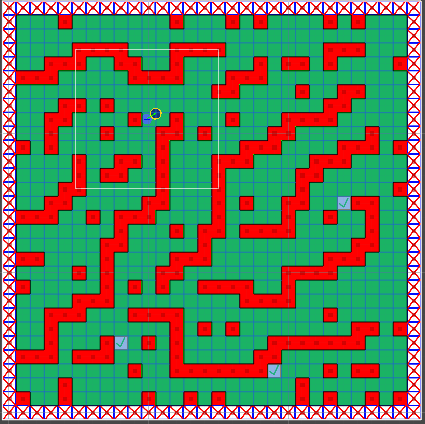
List the platform and software tools you plan on using

1. Stuff we will use:  
   Unity (with C# scripts)  
   Adobe Photoshop  
   Adobe Illustrator  
   MS Paint  
   Audacity

Screenshots below



Flowchart for level generation



A level generated by the algorithm