Project #3: Terrain Analysis (due Thurs, Week 8).

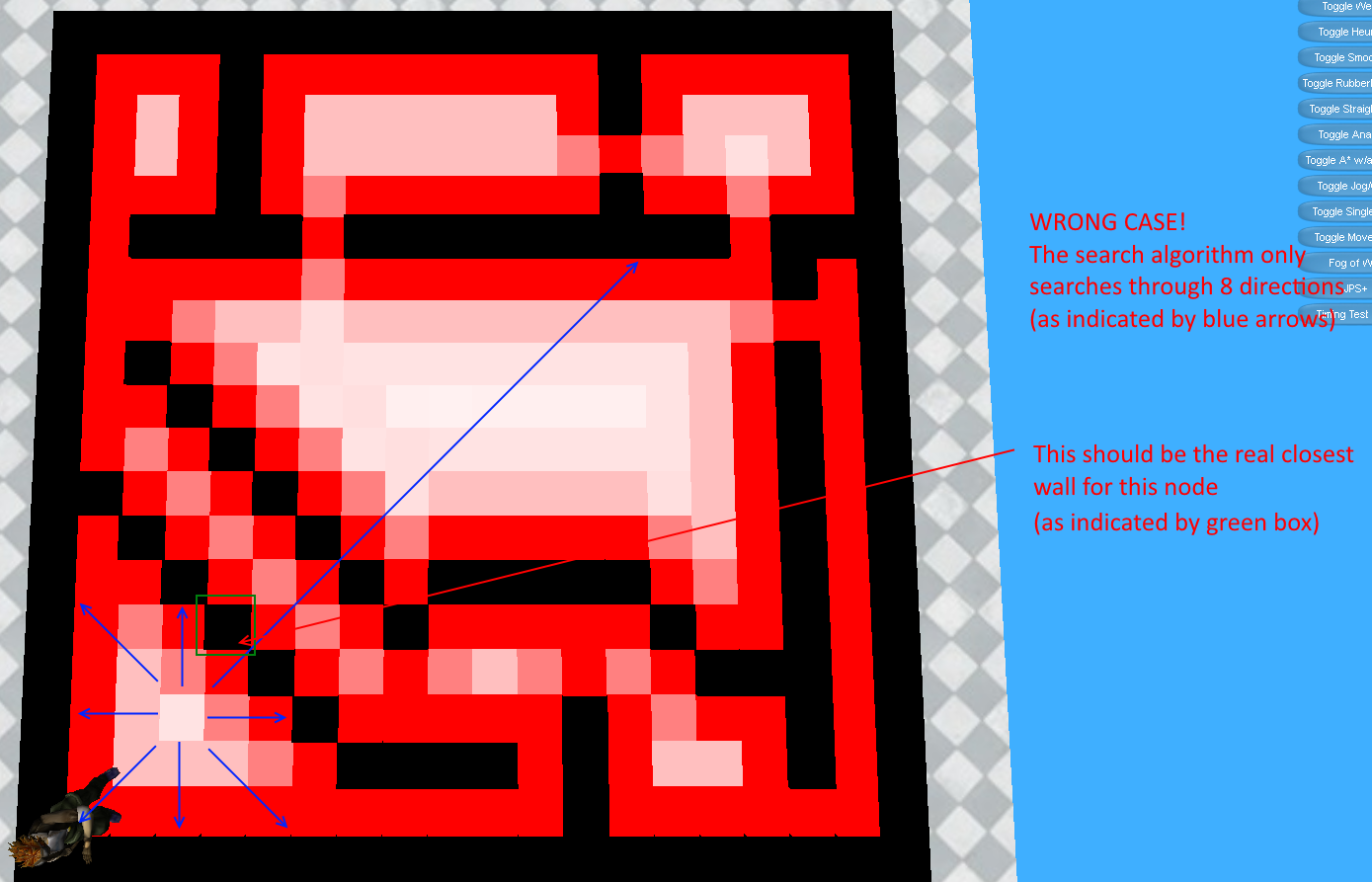
**Additional Notes:**

(Terrain Analysis)

1. Implement the following functions in terrain.cpp:
   1. ClosestWall
   2. RearCoverValue
   3. AnalyzeOpennessClosestWall
   4. AnalyzeVisibility
   5. AnalyzeRearCoverWithHighVisibility
   6. AnalyzeVisibleToPlayer
   7. AnalyzeSearch
   8. IsClearPath
2. Create more helper functions for your need.

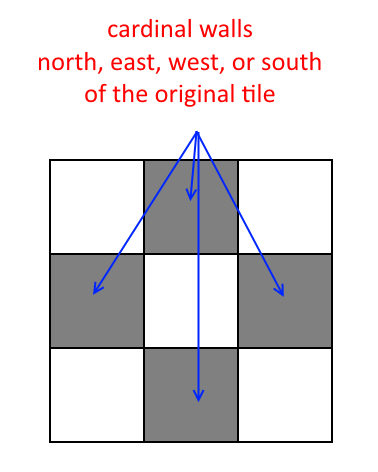
(Terrain Analysis : Openness Closest Wall)

1. One common mistake to find closet wall of a given grid is to only search for 8 directions. This method will miss the following case (as picture) where the closet wall is not at either horizontal/vertical/diagonal direction of the grid.

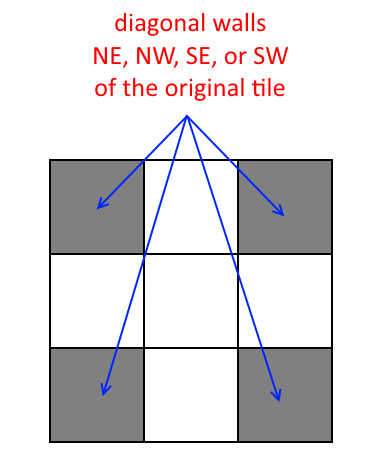


(Terrain Analysis : Openness Rear Cover)

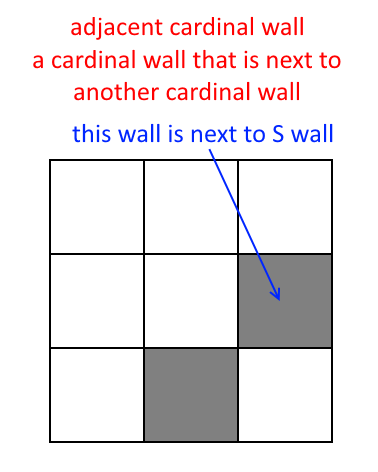
1. A "Cardinal Wall" is a wall that is North, East, West, or South of the original tile.



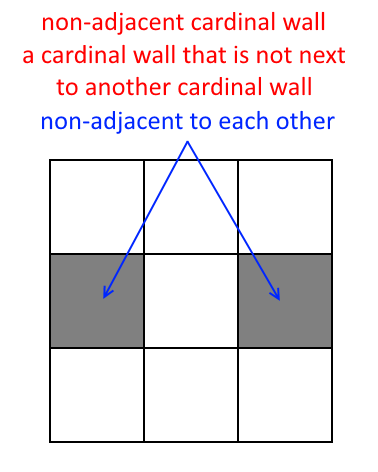
1. A "Diagonal Wall" is a wall that is NE, NW, SE, or SW of the original tile.



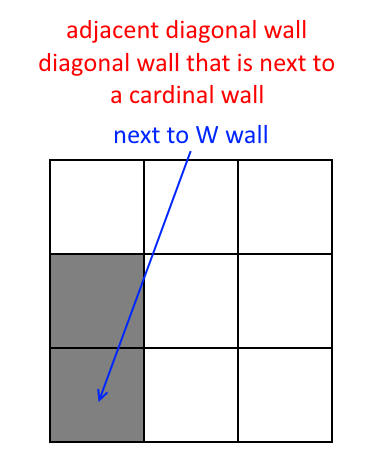
1. An "Adjacent Cardinal Wall" is a cardinal wall that is next to another cardinal wall.



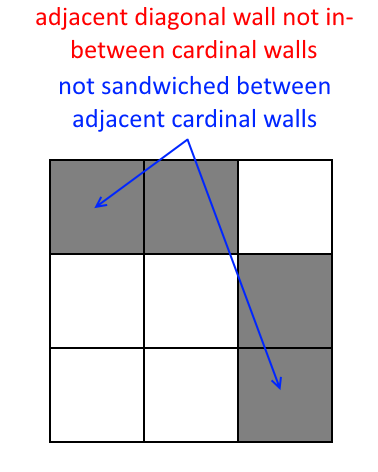
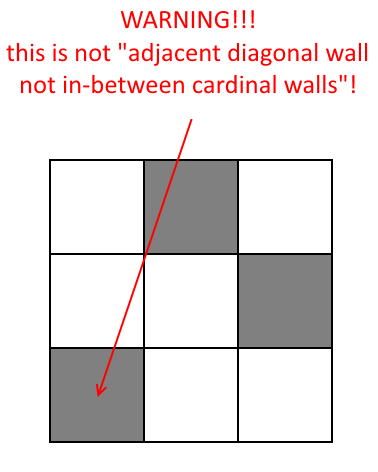
1. A "Non-Adjacent Cardinal Wall" is a cardinal wall that is not next to another cardinal wall.



1. An "Adjacent Diagonal Wall" is a diagonal wall that is next to a cardinal wall.

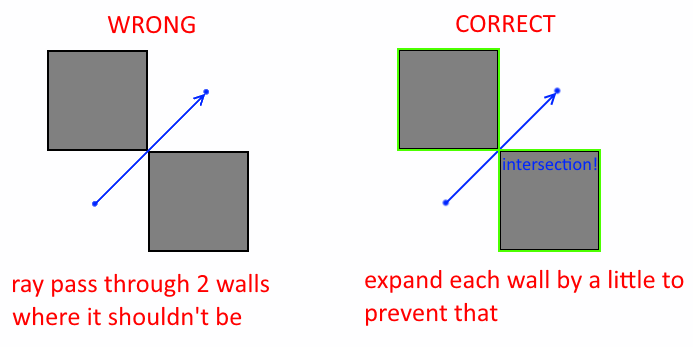


1. An "Adjacent Diagonal Wall Not In-Between Cardinal Walls" means an adjacent diagonal wall not sandwiched between two adjacent cardinal walls.

(Terrain Analysis : Visibility)

1. Use LineIntersect() function in terrain.cpp for ray casting.
2. For IsClearPath(). You need to add a small epsilon to prevent the following problem:



(Terrain Analysis : Visible to Player)

1. Clear previous analysis data (see example solution).
2. Two conditions of a sniper spot:
   1. It is next to 1.0 grid (the neighbor of such grid).
   2. It is visible to 1.0 grid (if you do raycast of the center of this grid and 1.0 grid, the line doesn’t intersect with walls.
3. Sniper spot should not pass through walls.



(Terrain Analysis : Search)

1. For field of view. Recall the dot product formula to get the angle of two vectors.
2. You do not need to get actual angle to check field of view in this project. What’s the special property in dot product if the angle is 180 degree?
3. Don’t forget to normalize vectors.

(Fog of War : Wall Visibility)

1. I will only test Fog of War with Terrain Analysis turned off. So you do not need to worry if it breaks with any of the Analysis feature.
2. Example solution is not 100% correct in this feature. So it is OK if your project does not produce the same result as the example. Tiny should only see walls that are not blocked by other walls and within the field of view, however.

(Fog of War : Path Validation)

1. As long as tiny does not pass through walls. It does not matter if you choose to re-path whenever a tiny sees a new wall, or only re-path if the current path passes through walls. It is easier to program the first way, though. For your reference, In TimeToBeat\_ExtraCredit30Percent\_Chi-Hao.exe there is a button "FoW Repath Method" to toggle between two ways of implementation.
2. Since tiny does not know invisible wall is wall until it sees that. It is possible to set the goal at invisible wall. In your project tiny should stop moving when it realizes the goal is wall.
3. Tiny should stop A\* search immediately when it realizes the goal is either not reachable or if the goal is wall (example solution does not take this into account, but you should).