# Game Concept

Our game is a computerized version of the classic tile-based game Carcassonne by Rio Grande Games. The rules for the game can be found at:

<http://www.riograndegames.com/uploads/Game/Game_48_gameRules.pdf>

The object of the game is to lay tiles on the playing field adjacent to existing tiles, and score the most points by the time all the tiles are used up by placing followers in cities, cloisters, and farms, and on roads.

Players take turns laying the top tile from the stack of remaining tiles. The player can choose where to place the tile and whether to place a follower on the tile.

# Planned Structure

Each player’s score will be displayed in a unique color on the side of the screen. A prompt will indicate which player’s turn it is. The game will be played using an isometric 3D camera which can scroll at the edges of the screen when the board grows too large to fit all tiles on screen. Additionally, the mouse wheel will allow the player to zoom in/out on areas of the board. While the player is placing a tile, the tile will appear hovering over the board semi-transparently. Possible tile placements will be shown in green, and adjacent tiles where a piece can’t be placed will be shown in red. The player can rotate the piece they are placing by right clicking. After a tile is placed, the player may have the option of placing one of his/her followers on the placed tile. Remaining followers are displayed in the corner of the screen for the current player. The game will automatically determine if a city/road/cloister has been completed and award points to the corresponding player.

We will build a main menu and a menu while in game to return to the main menu. The total number of games won for each player will be tracked, along with their current streak. A tutorial menu will explain the rules of the game. The tutorial menu can also be accessed from the in-game menu. When a new game is started, the names of the players are collected, along with whether they are human or AI players. If time permits, we may implement multiple AI difficulty levels. The game ends when all the tiles are used up (or the remaining tiles can’t be placed anywhere). The player with the highest score wins. A summary menu will pop up showing each player’s final score and the winning player.

# Workload Distribution

We plan to work together so that we can communicate more easily about any issues as they arise. Ben will work on the main engine code, and Josh will focus on implementing individual game objects and game logic, but we will split up and share tasks as we see fit while working.

# Anticipated Issues

Since we are using an isometric view, mouse collision detection will be somewhat more complex than simple top-down 2D collision detection. By using GLM’s matrix types and functions, we can calculate the inverse of the projection matrix to transform clip coordinates to world coordinates, calculating a ray from the camera in the direction of the mouse. The point where this ray intersects the game tile plane can then be used for 2D collision detection with individual tiles.

Creating a “dumb” AI should be trivial, but creating an AI that chooses the best from a set of possible placements for a tile or choosing whether or not to place a follower on a tile will be more difficult. We could implement a Case-based Reasoning system for the AI players which would start out “dumb” but gradually learn better strategies based on choices that led to scoring points. The AI could learn over the course of multiple games. Alternatively, we could simply implement heuristics which would be hard-coded to allow AI’s to try to pick good options.

Each tile can potentially be a part of numerous cities, farms, or roads, so we will need a way to keep track of which tiles compose different entities so that we can calculate when said entities are complete and score them correctly. We will probably have separate classes for farms, roads, and cities to facilitate this (separate from tile objects).