Alchemy is a more complex game than Tic-Tac-Toe, and in order to create it you will need to add several features to your engine. This week, you will start with a simplified version of the game that displays a world through a viewport.

During this checkpoint, you will be implementing a component-based design pattern in your engine, eliminating the need to hard-code any logic into one class. Be sure to go through the lecture slides thoroughly, and don’t be afraid to ping #questions for any clarifications!

We will be handling collisions next week, so don’t worry about how to determine whether your “units” are colliding just yet.

**Design Check**
- How will you add GameObjects to the relevant systems?
- How will you enforce a drawing order?
- How will you incorporate viewports into your existing UI toolkit?

**Primary Requirements**

**Primary Engine Requirements**
- Your handin must meet all global requirements.
- Your handin only crashes under exceptional circumstances (edge cases).
- The engine must have a “viewport” UI element that draws the GameWorld from a different coordinate system onto the screen coordinate system. The viewport must support panning (translation of the currently viewable game state).
- Your engine must have a System class that contains GameObjects.
• Your engine must have a class representing the game world. That class must hold at least one system.

Primary Game Requirements
• The viewport should be visible once the game is run (either directly or through a menu). The player must be able to pan the viewport (the viewport cannot display a solid color).
• There must be a “unit” visible in the viewport.
• The “unit” must respond to being clicked (only when the unit itself is clicked).
• The “unit” must be able to be dragged freely.
• The “unit” should be drawn and moved using separate components.

Secondary Requirements

Secondary Engine Requirements
• Your engine must meet all primary engine requirements.
• The viewport must support zooming (scaling of the currently viewable GameState). When zooming, the viewport must either keep the center of the game in the same position or keep the location currently under the mouse in the same position.

Secondary Game Requirements
• Your game must meet all primary game requirements.
• The player must be able to zoom the map (make sure the unit still responds to being clicked!).
• There must be a menu from which the player can make a copy of a “unit” by clicking and dragging it, without the original being destroyed or moved.

Suggested Extras
• Give your viewport limits on viewing parameters:
  ○ Don’t let the player zoom too far in/out.
  ○ Don’t let the player pan out of the view of the game.
• Give your viewport momentum-based panning and zooming, which will provide a smoother transition between two locations.
• Allow things to be drawn using coordinates in game space while maintaining their size (e.g., health bar that follows a unit without zooming in or out).
● Have your GraphicsSystem transform game space into GameObject space before passing the GraphicsContext2D object.

**Handing In**

Handing in works the same way as it did last week for Tic! Feel free to refer to the bottom of the [Tic assignment handout](#) for a walkthrough of each of these steps. Note that you do not have to join Gradescope again. To hand in...

1. Push your final handin commit
2. Create a release for this handin