**Abstract**
- Create a persistent world with arcade-style action gameplay without any dead reckoning.
- Design and build a game framework on top of DirectX and its helper libraries.

**System Overview**

**Networking**
- Initially attempted to use Quake 3 networking model
  - UDP for everything
  - Send position differences and use delta compression
  - All player state data is sent using UDP. Login and chat is sent via TCP. Putting these on UDP would force me to write the TCP protocol.
  - Exact position of each ship, hit points, and time stamp is sent to each client. The client orients each ship in the direction of its target, and sets the speed depending on how far in the future the timestamp is. Hit points are set immediately. Using state differences required the client to store a lot of past data received, past states on the server, and send a significantly greater amount of data to the server.
  - For the small amount of data sent, delta compression was not needed.
- Client knows of new ship when it sees an id it does not recognize
- Server identifies client input by IP Address and a random number sent with each input. Routers change UDP ports at any time without notice.

**Game Description**
- Gameplay is 2-D. Players can accelerate, decelerate, turn left, and turn right using WASD controls.
- You can see other players’ movement
- Graphics are 3-D.

**Heads-Up Display**
- Completely data driven using an XML file, and coordinates are in normalized coordinates, so the data is independent of screen resolution
- Data for HUD elements is set on each element between frames
- Components
  - Meter – A meter, in this case, used for health and energy
  - Text Label – A box that displays text, used to display speed and position
  - Rectangle – A textured rectangle, used for the background

**Graphical User Interface**
- Data driven and normalized coordinates like HUD
- Sensitive to keyboard and mouse input
- Input events are set in code
- Widgets
  - Textbox - a textbox that accepts alpha-numeric text
  - Button - a standard button
  - Label - a simple text display
  - Text Area - a multi-line textbox

**Conclusion**
- The gameplay is a bit jumpy when there’s a lot of maneuvering, but otherwise is smooth. Dead Reckoning would have fixed this. There is also a noticeable lag between input and the ship reacting. This is over the LAN of Harrison House.
- The game framework is quite flexible.
  - The low level components, such as rendering, input, and sound, can easily be swapped out for more featured APIs.
  - The Screen, GUI, HUD, and Entity management code can be used for future projects.