**Steve feedback notes:**

· Create a prototype with very basic functionality ASAP (start screen game screen, perhaps the ability to click and drag objects in game, exit to main screen, exit game) no need for any puzzles, just the skeletal game structure.

-Focus on getting GUI scaled for all devices

-Brainstorm mechanics

-See if we can get it so the canvas is different per platform

-Make objects interactable with both Mouse **and** touch

Justification:

-To demonstrate our ability to create our game for multiple platforms

-To reach a larger playerbase

o Then get that skeletal game onto Windows, Android, iOS ASAP. Work out all the intricacies of pushing the game to three platforms and decide quickly if you really want to aim for the three OS (that’s a lot of administrative overhead !). I would recommend Windows and Android at the most. I have published on iOS and it was unpleasant experience registering a company and getting the game approved.

-Work for Windows and Android for now

-iOS will come later if all goes smoothly and we find it to be in scope.

o Create a working copy of the game once per week and push it to your OS and check it on the device every week (check icon/button size, graphics/models are clear… do you need different assets for Windows and android ?) You will be taking the game from a large desktop monitor to a mobile phone, so do not expect graphics/models to look the same across both devices. \***PLEASE**\* do not leave this until the end of the project !

-”Always” have a *stable* build which runs on all ‘platforms’

-Keep A *test* and *stable* branch

· If you are suggesting that the player can invent their own solution, this suggests emergent gameplay which is going to take a heck of a lot of play testing, don’t expect your first implementation to be the best. It sounds as if paper prototyping is going to be very important here, as it will be a lot faster than coding changes, so break out the paper and scissors. I do not doubt the teams ability to get the code working but the ‘game design’ issues here are going to either make or break the game.

-Paper prototype ideas

-Rapid prototyping

· Think about how you will teach the player the rules of your game and how will you increase the difficulty at a rate that is not going to frustrate the player, again, this is more time consuming that many people appreciate.

* Either have tutorial levels to help the player learn core mechanics
* Or/And, use a mechanic that we can base a solid LPM loop around
* Or/And, make the difficulty scalable through code

-Make use of the LPM loop

-Once the player has learned a mechanic, use it in different ways and introduce new mechanics

· I would like to see the team taking time to deconstruct other similar games (which you have referenced) and break them down so you understand all the details. Also take time to design an engaging interface. Take time to break down the interfaces for the games you referenced. For example, look at the font which Poly bridge has used (why that font and not just Time New Roman?. Notice how the graphical style of the water matches the style of the land terrain. In SpaceChem, notice the grid squares have rounded corners, not just standard sharp corners, as do the boxes in the GUI at the bottom of the screen. Why do think the two larger boxes on the right edge of the GUI have a lighter background than the smaller boxes in the middle of the GUI? It’s this kind of detail and thought process I would encourage the team to focus on. Often, programming students focus so much on the code, that they either forget or do not even consider the importance of the user experience. This could be a weakness for the team as you are all programmers, but if you know that might be the case, then you can focus on it more. Do Not spend all your time on the code