600.112: Intro Programming for Scientists and Engineers

Assignment 8: Hacking a Game[*]

Peter H. Fröhlich
phf@cs.jhu.edu

Joanne Selinski
joanne@cs.jhu.edu

Due Dates: 11:30pm on Fri 11/6 and Mon 11/16

Introduction

The eighth programming assignment for *600.112: Introductory Programming for Scientists and Engineers* is very different from most of the previous assignments. We're not doing anything very scientific or very engineered at all. Instead we'll start winding down the semester with something a bit more personal: you'll hack a video game. To make things more interesting and (hopefully) enable you to develop more complex games than you could on your own, *you will be working with a partner on this assignment!*

This handout itself is not particularly informative since the project is pretty open ended. The one thing that **is** still very serious is this: As always, you must submit a complete zip of your project files on Blackboard before the deadline. Because you will be working in pairs, only one person of the pair needs to submit the completed assignment. *However, both partners must submit a textbox on Blackboard that states the full name and Blackboard ID of their partner.* In addition, the complete zip must include a plain text README file which details who worked on the project, why you choose the game you developed, why you designed your solution they way you did, and what pitfalls you encountered along the way. More details are listed below in sections 2 and 3. You can **lose points** if you create more work than necessary for the graders by not following the instructions.

1 The Game Options

You need to find a partner since this is a team assignment, and you can use a Piazza post to do that and also to register your team. **If you are taking the lab course, you must choose a partner in your section so that you can work together there.** You must use the paired programming method to develop your game. In true paired programming both partners work together to develop all the code - taking turns being the driver (hands on keyboard) and navigator (eyes on the road, directing the driver). Also make sure that you are sharing code regularly with each other so that you both have the most up to date versions of all your files. What not to do is try to divide up the work and program different parts independently. That is very likely to end in disaster when you try to integrate the parts.

Make sure that both partners fully understand the examples we do in class and the PYGAME code in your solution. You should **not** copy/paste them without understanding; there **will** be questions on

[*]Disclaimer: This is *not* a course in physics or biology or epidemiology or even mathematics. Our exposition of the science behind the projects cuts corners whenever we can do so without lying outright. We are *not* trying to teach you anything but computer science!

PYGAME on quiz3 and the final, and if you don't understand what you did for this assignment, you won't be able to answer those questions very well. You have been warned.

You can choose among four classic games to implement: Pong, Breakout, Snake, and Missile Command. Here are links describing the games you're allowed to build for this assignment without asking for permission.

Pong: Minimally you should build a single-player Pong game; however we encourage you to build the two-player version instead. Pong is not really challenging when played with a mouse, so try to use keyboard input instead.

Breakout: Similar to Pong, Breakout is probably too easy when played with a mouse. Ideally you use at least different colors for different layers of bricks, even better if you actually find and use some cooler graphics/sprites. A very cool example of an "advanced Breakout" with better graphics and even power-ups is Arkanoid and variants thereof.

Snake: The simplest variation of this will be a one player game without obstacles. Try to use unexpected graphics/sprites and integrate color changes to make things more interesting. Snake can be made more challenging to play by creating multiple levels with increasing obstacles or adding two players.

Missile Command: The most complex game of the bunch, but also the most fun perhaps. Note that you're free to replace missiles with bugs and cities with carrots and explosions with bug spray if you want a less "Cold War" feeling game. Make sure you limit the number of explosions that can be on the screen at the same time, and make sure you allow the user to play with the mouse (the game is too hard with a keyboard in our humble opinion).

If you don't like any of these games, feel free to ask for permission to do another game. Post a question on Piazza suggesting another game. Please include a relevant link so we can figure out if it's doable in the time you have. We need to give explicit permission for any game that's not the above four; don't start working on a different game until you get permission.

2 Part A: Game Specification and Design [20 points]

Recall the first two steps in program development: problem analysis and solution design. Those are the steps you need to complete and submit for the first part of this assignment. Rather than submitting any code, instead you will write and submit a plain text README document for this part of the assignment. (This will be the start of the README document that you submit with your final project.) Only one partner needs to submit the actual document, but both must submit a textbox on Blackboard that says who they are working with.

First of all, list the full names, Blackboard IDs, emails and phone numbers of the developers on this project. Next tell us which game you will be implementing, what particular features or enhancements you hope to give it, and why you made those choices. Be very clear about the expected behaviour of the game - 1 player or 2, visual enhancements, keyboard or mouse input, etc. These are the specifications that will guide your development. This completes the first phase of development.

For the second phase, you must determine which features of pygame you will be using, and what custom classes you will be creating. You should list the pygame features along with the components of the game for which they'll be used in statements something like this: From PYGAME we will use Rects to represent (blah) and (something) to control (whatever). The more details you provide, including why you made those choices the better. You should also determine what custom classes you will create (possibly inheriting from existing PYGAME classes). For each of these, state their names and a description of their purpose, as well as the primary data members and methods they will contain. Consider this to be a rough outline of the code you will create. Lastly, include pseudocode for any methods that will be algorithmically complex.

Ideally you will have this part of the assignment completed as soon as possible, so that you can move on to the coding. However, do not try to pass over this step too quickly - the more thorough you are here in thinking through your specifications and coding approach, the easier it will be to build a great game in the coding phase.

3 Part B: Game Implementation [70 points]

As much as we'd all like there to be detailed, specific, itemized criteria defining the correctness of every program we write, it often happens in the Real World that we're given instructions exactly as specific as "you need to write a game; go!" So this will be a good chance for you to demonstrate not just your programming know-how, but your design savvy and prioritization skills.

Some things we'll consider when grading your games, roughly in order of importance:

Does it run? Your program should show that you've playtested it enough that it won't crash during reasonably normal gameplay.

Does it play well? Did you implement a recognizable version of the game you're using for a basis? Do the controls and the behaviour of the game objects match up to what a player expects?

Is it well-coded? This is your opportunity to show off what you've learned this semester. Is your program organized in a sensible way? Do you put commonly repeated code into functions, or did you just copy and paste the same lines again and again? If you have to deal with a bunch of objects, are they in a list? a set? do you iterate through them with a loop, or write the same function call explicitly for each one? There's no one right way to write any of these games, but a good program is always readable, organized, and well-documented.

Did you spend some time getting to know pygame? The library can do a lot for you – sprite collisions, sound effects, ... Going beyond the example code will make your game more interesting, and demonstrate that you're comfortable reading documentation and using other people's modules (which we know you are – you've been doing it all term!)

Is it polished? Did you sit down and figure out why it looks like the ball bounces off the paddle just a tiny bit before it hits it? Did you tweak the rect object so that the collision is calculated to occur based on where the ball actually is, in spite of a little transparent space around it in the image you used? The core mechanics are more important, but little refinements are the difference between an acceptable coding exercise and an enjoyable game.

All the bells and whistles. Hopefully you're so engrossed by this project and programming in general that once you get rolling, you can't stop – you want Super Missile Command 112 to have on-screen tutorials, high-score lists, configurable audio and video options, 256 procedurally generated levels, 7.1 channel dynamic surround sound, 16 networked players, [...]. We'd love to see all of this of course, but you can make a perfectly respectable and fun game by just keeping the above points in mind. ***Significant extra features (ie, beyond the minimum for standard game play) may be awarded some extra credit points, with a maximum of 20 possible.***

The most important thing you can do for your grade (apart from writing an excellent program) is to **USE YOUR README**. All those hints up there in the form of questions shouldn't be rhetorical – you should answer them and similar questions to explain why your program is written the way it is. If it's perfect, tell us how it got that way. If there are things that aren't quite right, tell us how you struggled with them. If you ran out of time before adding a cool features, tell us how you'd implement it – and how your existing program architecture would make this a cinch.

If your game does a reasonable proportion of what you set out to make it do; if it plays consistently and intelligibly; if the code makes it obvious that you've been paying attention this semester; if it's evident you put the best of your programming ability into making it interesting; and if you provide a well-reasoned discussion of your development process and its result, you will get a good grade on this assignment.