Programming Paradigms – PyGame + Twisted, Spring 2016

Summary: Create a multiplayer game that allows a minimum of two players to compete (or cooperate!) via a network connection. The network architecture may be direct (peer-to-peer), or may use a server. You must use the PyGame and Twisted libraries.

Teams: This is a paired assignment, with teams of **two**. Of course, work closely with your teammate. You *may not* share code with other teams.

Grading: This assignment will be graded out of 70 points. You will get an individual grade, but these grades will be tied to your combined turn-in. I will assess these grades based on the following:

Effective use of PyGame library:	15 points
Effective use of Twisted library:	15 points
Scale and complexity of project:	20 points
Meaningful README describing gameplay:	5 points
Teammate + self evaluation:	15 points (or more, see caveat below)
	70 points total

Expectations:

"Effective use of PyGame library"

This section is an evaluation of the game *functionality*, independent of the gameplay. I will be looking for clean and readable code, use of the clock-tick design pattern, and use of the PyGame library functionality (learn the library, do not reinvent the wheel). If you learn from documentation or online examples, label those in your code. Keep incremental versions to show evidence of ongoing progress (for partial credit if your project ends "suboptimally"). Comment comment comment! ☺

"Effective use of Twisted library"

Similar to use of PyGame library, I will be looking for correct code strategies (mark evidence in the comments about why you made decisions and where you found examples). Take the time to learn the Twisted library, rather than trying to duplicate functionality. Integrate Twisted and PyGame using strategies from class and/or backed up by documentation or rationale. I will look for evidence that you know *why* you make the decisions you do. **Comment your code** and use your intuition about what are good coding decisions.

"Scale and complexity of project"

Ambitious projects with many features and fine-tuned gameplay will receive an A. Windows 95 Solitaire would receive an F. Ideas for being ambitious include: computer-controlled players, multiple multiplayer modes, highly customizable characters, carefully controlled gameplay, etc. There are many paths to an A! You could make a case that your network architecture is highly advanced, that your gameplay is very advanced with many complex NPCs and items, that you integrated a realistic physics engine. I'm looking for evidence of a good-faith effort to employ your creativity and interests.

"Meaningful README describing gameplay"

Describe the main features of your game. A written tutorial is acceptable, but an in-game tutorial is also acceptable. If you do an in-game tutorial, say how to get to that tutorial from the README. *Be sure* to explain exactly how to set up and start your game; imagine you are writing directions for a computer novice.

"Teammate + self evaluation"

Teamwork is essential here. Work hard for your team, but be careful not to dominate the project. Communication with your teammate is key. At the end of the project, prepare a ~1 page evaluation of yourself **and** your teammate. Send this evaluation directly to me via email. It is confidential: do **not** share it with your teammate. Explain what grade you feel you should receive and why. For your teammate, do not give a letter grade, but instead write your assessment of your teammate's strengths and areas of improvement. If your teammate avoided you or did little work, this is the place to say so.

Teammate evaluations will influence the grade you receive on this component. In the ideal case, both teammates will work together and share the labor, and write short, strong evaluations. I expect this to be the vast majority! In the "suboptimal" case, I will investigate any discrepancies in the team evaluations and assign points accordingly. You will not receive less than 100% on this section without hearing from me first. If I have evidence that a team member made little to no contribution, I reserve the right to lower their grade beyond the teammate evaluation.

Turn-in:
As always, package your assignment and send it to prog.paradigms.secN.sp16@gmail.com, where N is your section number. If your file is zilla-sized due to graphics or sounds, put it in Box and provide a download link, or make other arrangements to transfer the files to the professor. It is your responsibility to make whatever arrangements you need to make to get the files to us on time.

Learning objectives:
What you're really practicing here is the clock-tick design paradigm. Clock-tick is a very popular way of handling programs that have mixed automated and interactive elements. It is also useful in programs where many semi-autonomous components need to be synchronized to some degree of precision, as in distributed sensor networks or control systems.

You are also practicing collaborative software development. Very few programs are commercially developed by only one person. Software companies place a high value on team courtesy and knowledge of the division of labor. Oftentimes, the hard work comes during integration, when software components written by different persons are stitched together into a final product.

Final Due Date: May 4th, 7am EST (Grades are due May 9th, and we need at least three business days to give them all a fair evaluation.) You MUST turn in your assignment on time! Assignments turned in late will receive zero points!

Have fun and do good work! ☺