# TECHNICAL DESIGN DOCUMENT

The requirements for the Technical Design Document are listed below. All listed sections must be included, and they must be in the order listed. Every page must have a page number (except for the cover page), your team name, and the DigiPen copyright. Spell-check and grammar-check the document before you turn it in. Documents must be neatly formatted and easily readable. Put page breaks before new sections (when appropriate), use consistent formatting and fonts, use headings for sections and sub-sections, etc. Sloppy documents are unprofessional and will prevent your project from getting funding.

While you should try to think through as much of your technical design as possible in this document, don’t worry about getting it perfect or entirely fleshed out. In particular, you should focus on what technology you need for a really good first playable. Any technology that isn’t critical to a good first playable should be labeled as such (make these things a different color, put them in italics, or whatever works for you).

This is the document that will tell every programmer on your team what they need to know to really start implementing the game. If you suddenly added a new programmer to your team, this should be the document you hand them so they can get familiar with your engine and start being productive as quickly as possible. If you have a really good engine proof, your TDD will essentially just describe what you have already done, but in most cases you will still have some areas where you will have to describe what you plan to do instead of what you have done. This document is usually about 15-25 pages long, but could be less for a simple game.

**TDD Structure**

**Cover Page:** Game title, team name, class name and section (GAM300A), semester, and year. Also list all members of your team, their official job (or jobs) and their coding responsibilities. Game design and art students should also be listed (and noted as such). Yes, all of this goes on one page.

**Table of Contents:** You must have a TOC—make sure you update it before you turn it in.

**Overview:** Describe the overall architecture of the project. List and describe each global component the project uses (graphics, physics, networking, AI, UI, file I/O, memory manager, audio, etc.). Describe how each global component is updated when the game is running and how they communicate with each other. List and give a brief description of all the major object components in the project. Describe how these components are made part of a composition. Describe how one component interacts with another component in the same composition. Describe how your objects are created/managed/destroyed/serialized. If multi-threading is going to be used in the game, make sure to note the implications of this to any relevant sub-system or composition.

**Graphics Implementation:** What techniques will be used to implement the graphics design? What graphics API will you use? Are you using fixed function or shaders? How are you loading assets such as sprites, models, textures, and animations? List everything you intend to implement, even if you aren’t sure what will really work at this point.

**Behavior Implementation:** What types of algorithms will be used to implement the behavior design? Pattern movement? Pathfinding? State machines? Flocking? Influence maps? Genetic algorithms? List everything you intend to implement, and what it will be used for, even if you aren’t sure what will really work at this point.

**Physics Implementation:** What techniques will be used to implement the physics design? What type of integration will you use (Euler, Improved Euler, Verlet, Runge-Kutta)? What type of space partitioning will you use? What type of collision detection will you use? List everything you intend to implement, even if you aren’t sure what will really work at this point.

**Multiplayer Implementation:** What techniques will be used to implement multiplayer? If your game is networked, how will you host games and detect players? What networking protocols will you use? Will you use encryption, data compression, host migration, etc.? List everything you intend to implement, even if you aren’t sure what will really work at this point.

**Coding Methods:** Describe all coding conventions that are to be used on the project, including file naming conventions, file locations (i.e., which files go in which folders in the project), code formatting, and code documentation. Also describe what kind of source control system you will use and any rules your team has about its use.

**Debugging:** Describe the support you have for debugging in your game. Do you have an in-game debug console (or at least a simple output-only console)? Do you have a debug drawing system? Do you have a clean assertion system? Do you have an in-game performance viewer? Do you have a way to watch variables in-game? Can you shut off your debug controls?

**Tools:** Describe any tools that need to be created. This includes map editors, art and audio processing tools, testing tools, etc. Also describe any scripting languages you have integrated into the game, including how it is bound to C++ and what it is used for.

If there are aspects of your design that do not fit in to any of the categories listed here, create your own appendices for that material.

**Appendix A**

**Tools and Editor Instructions:** You must include specific instructions on how to launch and use the tools for your game. These instructions should be written for the designer or content creator who will be using the tools. If you include a Content or Level Editor, you must include specific step-by-step instructions for its usage.