**Sample Proposal**

http://www.writing.eng.vt.edu/workbooks/proposal.samples.html

**Introduction**

Unity3D

Cocos2D

Stencyl

STATE CHANGE

AI

http://www.altdevblogaday.com/2011/02/24/introduction-to-behavior-trees/

SYSTEM DESIGN

-prototypes  
<http://programming-musings.org/2006/02/05/beyond-mainstream-object-oriented-programming/>

This allows for programs that run indefinitely --- associated with realtime programming

-Component based architecture.

<http://www.raywenderlich.com/24878/introduction-to-component-based-architecture-in-games>

The basic idea behind component based architecture is to **prefer composition over inheritance**.

This is a fancy way of saying “instead of inheriting from an object to reuse code/get functionality, build your game objects out of sub-components instead.

**Note:** Component based architecture also helps you do some quite fancy things. Instead of hard coding what components are in each game object, you could read a definition from an external file instead. This makes it really easy to create different behavior for your game objects without having to code, which is great for designers and quickly tweaking/iterating your game design.

* ***Pros:*** Flexible approach with a nice separation between data and systems
* ***Cons:*** Code can get a bit long-winded and tedious at times compared to simpler approaches

#### 1.1 Limits of object-oriented programming

<http://www-inf.it-sudparis.eu/cours/mari/EnLigne/Cours/ComponentMW/2.1.h>

http://www.saylor.org/site/wp-content/uploads/2013/02/CS101-2.1.2-AdvantagesDisadvantagesOfOOP-FINAL.pdf

# Software architecture

<http://www.spica.com/products/software-architecture.asp>

<http://www.cs.ccsu.edu/~stan/classes/CS530/Slides/SE-19.pdf>

Evolution support

There is a trade--off between reusability and usability

•• The more general the interface, the greater the reusability but it is then more it is then more

complex and hence less usable.

• Add a configuration interface for component adaptation.

• Integrate required components to reduce dependencies.

<http://www.cs.ubc.ca/~ycoady/acp4is03/papers/clemente.pdf>

<http://arxiv.org/ftp/arxiv/papers/1011/1011.2163.pdf>

<http://www.triz-journal.com/archives/2002/03/b/>

<http://www.sei.cmu.edu/architecture/research/previousresearch/lifecycle.cfm>

<http://userpages.umbc.edu/~khoo/survey2.html>

Top-Down/ Bottom-Up Design: Top-down design directs designers to start with a top-level description of a system and then refine this view step by step.

Stepwise Refinement: The iterative process where each system is decomposed and refined step by step is called stepwise refinement.

Structured Design: It is a data flow-oriented design approach. SD views systems from two perspectives : as the flow of data and the transformations that data flow undergo through a system.

Structured Analysis and Design Technique: The SADT methodology provides a precise and concise representation scheme and a set of techniques to graphically define complex system requirement

Jackson Systems Development: The JSD methodology models the world in terms of entity-action-attribute, which undergoes a step by step process to connect it to the "real" world

Structured Systems Development: SSD is based on the data structure-oriented design approach where the focus is on the output and the processes that transforms the input data structures to the output data structures.

Object Oriented Design:Object Oriented Design (OOD) provides a mechanism that encompasses three important concepts in software design : modularity, abstraction, and encapsulation (also called information-hiding).