GameSample represent the realization of the Component Entity System pattern.

The small game shows the functionality. It has field with walls, coin , bomb and player. The goal is to eat the coin and not touch the bomb. To control the player keyboard’s up, down, left, right keys are used. You can find the code that creates the game in GGame class.

All entities in the game are sets of components. Entity just the numeric identifier, that is used for getting access to components.

Creation of entites and component are managed by gEntityManager. CreateEntity() method creates and returns the Entity.

Template method AddComponentsToEntity(...) creates the component, that was passed as the typename for specified entity.

Component is stored in ComponentPool. Each Component has own memory pool. ComponentPool provides a resizable, semi-continues and cache-firendly memory pool for creating, storing and destroying components.

ComponentPool doesn't free memory after deleting the component, instead it reuses that memory for new components. Each ComponentPool knows what Entity is owner of the component. Also it stores the vector of component indexes for each entity. Using this vector we can get component for particular entity.

All Component are processed by system. And each component has system to be processed . For example, RenderableComponent has the sprite to be drawn . RenderSystem process all this type of component in update() and draws their sprites.

Systems are managed by SystemManager. It provides functionality for registering and updating the systems.

Resources:   
All sprites are packed to a texture atlas in png format. An information about sprites are stored in xml config file. We have 2 classes for managing the sprites and textures.   
GResManager class parses the xml config file, creates the sprites and puts them to hashTable(gResourceDictionary class). gResourceDictionary uses the name of the sprite to get the hash using the jbd2 algorithm.

gTextureManager class loads and unloads textures. All textures are stored in gTextureDictionary , that works similar to gResourceDictionary.

Communication between different component is provided by signal and slot systems like in QT([sigslot - C++ Signal/Slot Library](http://sigslot.sourceforge.net/)). Components contain signals