# Game Class Library

* Light\LightsFX.cs – Responsible for painting shadows over an image
* Light\LightSource.cs – Draws initial shadows
* Light\QuadRenderComponent.cs – Renders shadows as geometry
* Light\ShadowCasterMap.cs – Generates a map of shadow objects
* Light\ShadowMapReductionStep.cs – Contains information about how reduction is applied to shadows
* Light\ShadowMapResolver.cs – Resolves shadows and applies post-render effects like attenuation and blur
* AudioZone.cs – Information about an area where a certain song is played. Not used in our demonstration
* bytes.cs – Conversion methods for transforming common data types to bytes
* Enemy.cs – Information about enemy players. Not used in our demonstration
* EnemyAudioZone.cs – Info about areas near enemies, where we need to play menacing sounds. Not used in our demonstration
* enums.cs – Enumerated types used throughout other classes
* Extensions.cs – Extensions used for:
  + Converting int to bool
  + Testing for the existence of a point inside a rectangle
  + Position objects on the game map
  + Swapping elements in an array
  + Refreshing list viewer controls
  + Rendering text, rectangles and radial gradients
  + Writing common data types to a stream
* game.bmap – The game map. Stores all map properties, including map metadata and graphic references to indices in the tileset
* game.tileset – The game tile set. Stores the raw images and metadata used in the game map
* Map.cs – The map object. Contains all tile and map info, and methods for interacting with the map
* MySqlConnector.cs – A class used to connect to a MySql database. Not used in our demonstration.
* Player.cs – The player object. Contains all positioning information, and references to graphic indices, as well as methods for controlling movement
* Tiles.cs – All classes for different tile types
* Tileset.cs – The tileset object. Contains methods to load and save graphic and tile metadata information

# Server Class Library

* Chat.cs – A class with information about colors and data in chat messages
* client.cs – Creates and maintains a connection to a server. Used to communicate with other clients
* config.cs – A class containing methods to load from, save to, and interact with configuration files
* ConnectedPlayer.cs – A child class of Player. Contains information about the connection status of a player to the server. Receives/sends and handles messages from the server
* database.cs – A class containing methods for interacting with a remote database
* enums.cs – Enumerated values used by the server
* exceptions.cs – Exception classes used by the server
* GFXLoader.cs – Used to load graphics from a folder relative to the application. Not used in our demonstration.
* packet.cs – The packet class. Contains methods for adding and interacting with data. Instance of this class are what are sent to and from the server
* packetstruct.cs – Contains overloaded packet classes used for sending specific types of information
* server.cs – A partial implementation of the server. Contains methods for:
  + Starting a server
  + Maintaining the server
  + Accepting connections from client
  + Disconnecting clients
  + Sending packets to clients
* server\_handler.cs – The other half of the implementation of the server class. Provides methods for handling messages received from clients and responding to them as necessary
* World.cs – A static class with information about the players, map(s), lights, and game state

# Game Client

* AnglerGame.cs – The main game class. Contains all game properties, as well as the framework which handles updating, loading, and rendering the game client
* AudioManager.cs – A class used to manage transitioning and playing audio
* Background.cs – Used to render an image stretched across the expanse of the entire background
* BackgroundImageRenderer.cs – Used to render a background image
* ChatRenderer.cs – Renders and updates a chat window and all chat messages
* FXCollection.cs – Contains all graphics, songs and sound effects
* GraphicEngine.cs – A generalized class for rendering graphics objects
* InputHandler.cs – Handles input while the client is in-game
* LoopedSoundPlayer.cs – A sound player which loops output of a .wav file
* MapRenderer.cs – Renders the map, with shadows
* MinorPlayerRenderer.cs – Renders and updates all non-main players
* OverlayRenderer.cs – Renders and updates all overlaid elements – the light circle, bacteria count, etc
* PauseMenuRenderer.cs – Renders and updates the pause menu
* PlayerComponent.cs – Renders the main player
* Program.cs – The main entry point for the client application
* RippleEffect.cs – Updates and renders a ripple effect over a background image
* ShadowCaster.cs – Contains methods to draw shadows using the shadow classes in the Game Class Library

# Server

* Program.cs – The main entry point for the server application. This class starts a server instance and accepts administrative commands

# XNA Controls

* XNAButton.cs – An XNA button control
* XNAControl.cs – A generalized XNA control
* XNADialog.cs – An XNA dialog control
* XNAHyperLink.cs – An XNA hyperlink control
* XNALabel.cs – An XNA label control
* XNAMenu.cs – An XNA menu control
* XNAMenuItem.cs – An XNA menu item control
* XNAPanel.cs – An XNA panel control
* XNAPictureBox.cs – An XNA picture box control
* XNATextBox.cs – An XNA textbox control

# Game Content

* [multiple files] – Images, sounds, and effects used in the game