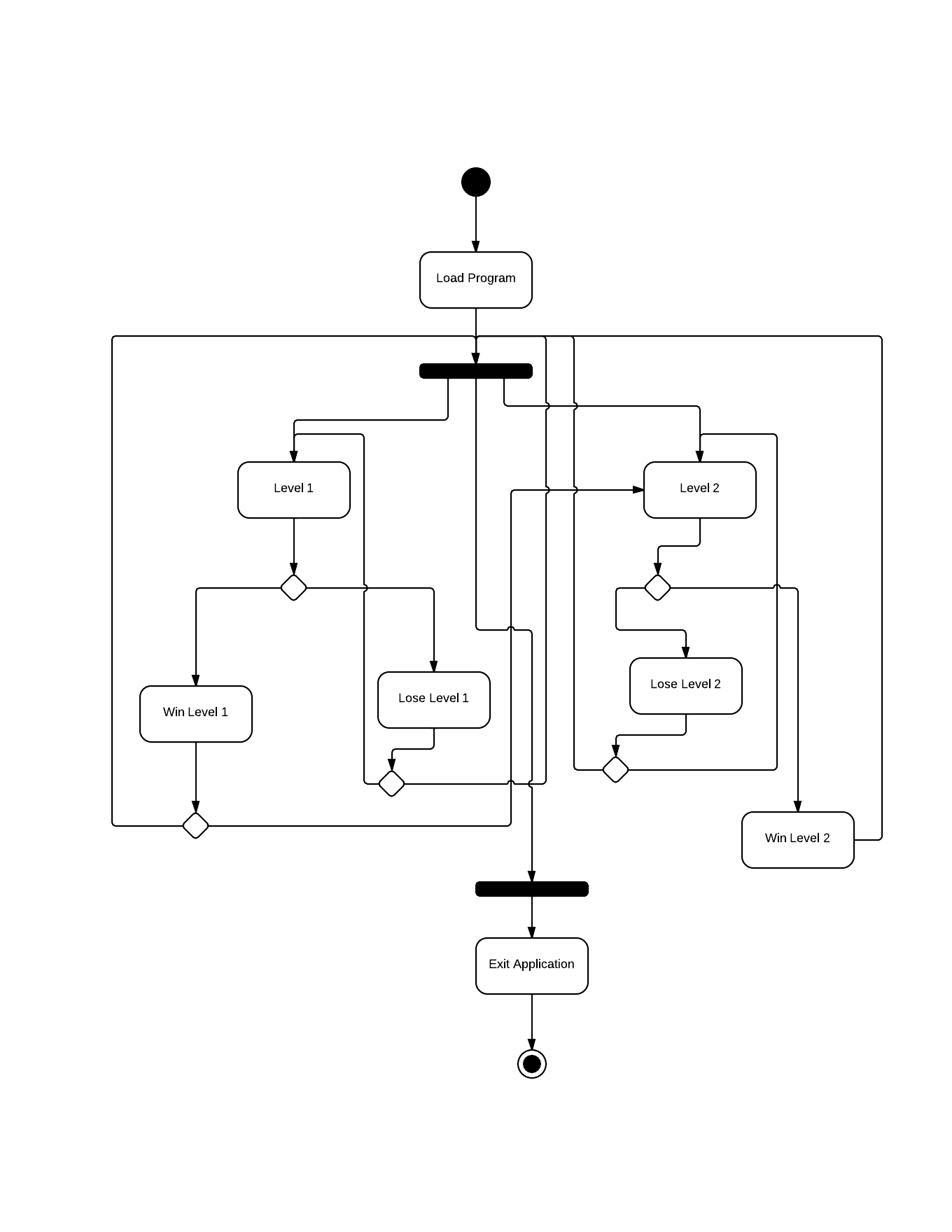
Technical Design

# Game Mechanics

* Created and tested on PC, works on Xbox
* Not tested on any other devices
* No external libraries used
  + http://xbox.create.msdn.com/en-US/education/catalog/sample/game\_state\_management used as a guide for Game State Management

## Whole project Activity Diagram:

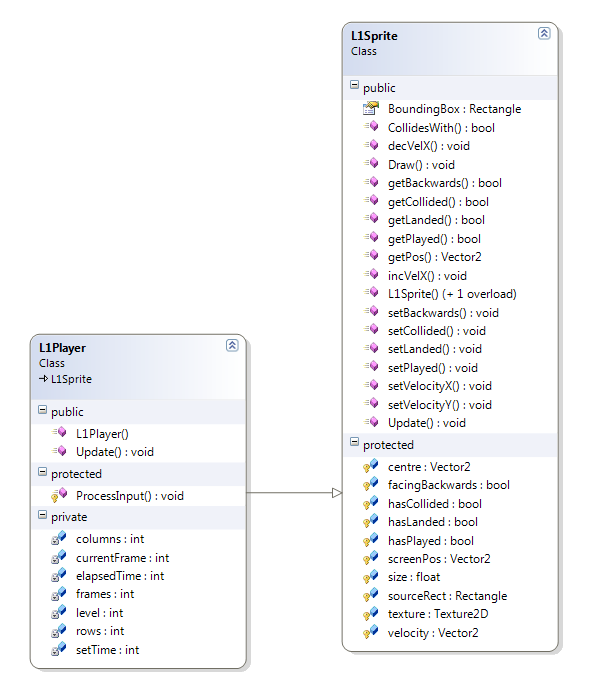


## Whole scale UML Class Diagram:C:\Users\Aaron\Documents\GitHub\SnowyCastle\UML\UML Class Diagrams\overall closed nodes.png

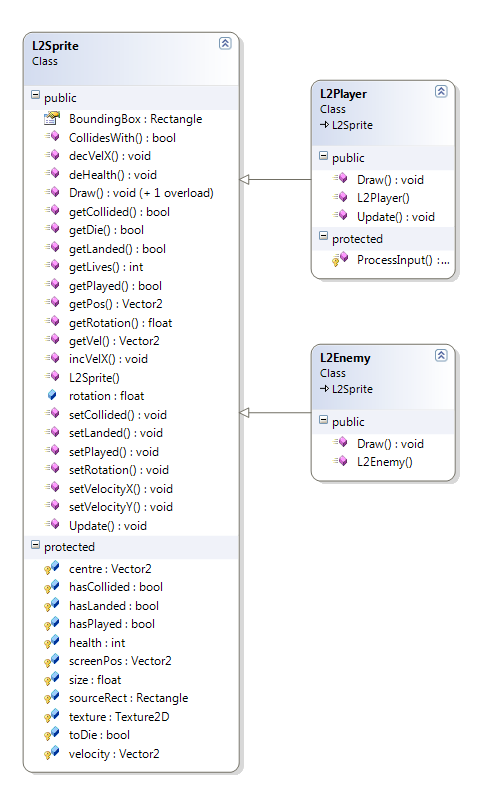
## Level 1 and 2 UML Class Diagrams

## C:\Users\Aaron\Documents\GitHub\SnowyCastle\UML\UML Class Diagrams\level 1 and level 2 uml class diagram.png

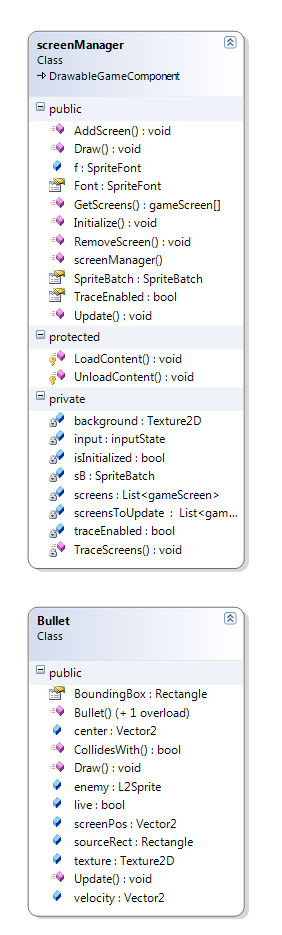
## Level 1 Sprite and Player Inheritance UML Class Diagram



## Level 2 Sprite, Enemy and Player Inheritance UML Class Diagram

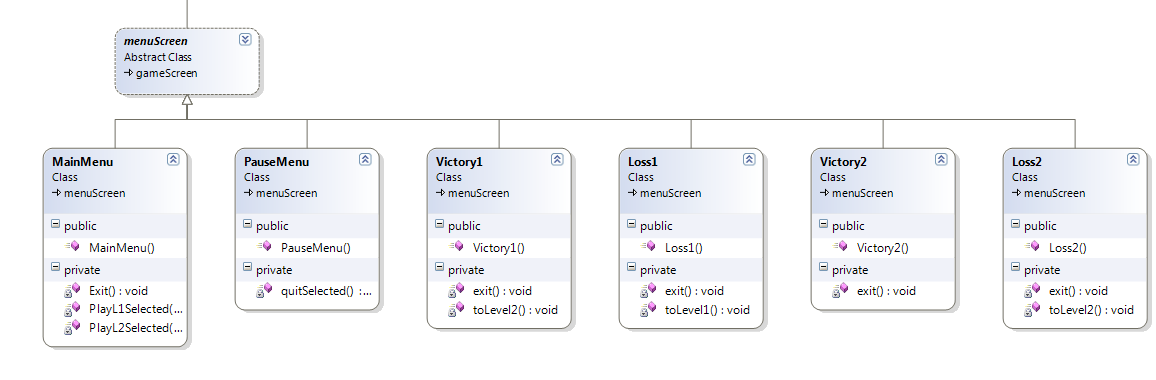


## Bullet and Screen Manager UML Class Diagram

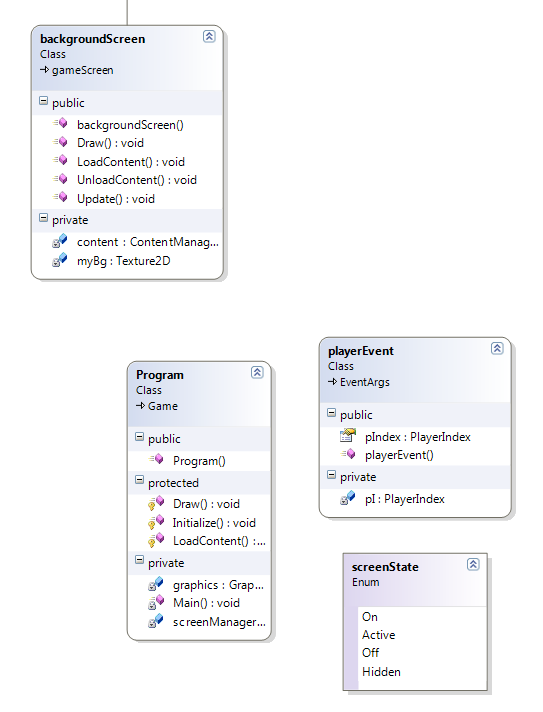


## Menu Item, Background, and Input UML Class DiagramC:\Users\Aaron\Documents\GitHub\SnowyCastle\UML\UML Class Diagrams\menu item, background, and input uml class diagram.png

## Pause, Victories and Losses Screens UML Class Diagram



## Program, Event, Screen State enumeration, and Background Screen UML Class Diagram



## Collections

* Level 1 has 3 Lists of L1Sprites
  + One for all the live snowballs
  + One for all the snowballs that have hit the ground and are inactive
  + One for the snowballs that have hit the player and need to be deleted from the main list
* Level 2 has 7 Lists; 3 of L2Enemy, and 4 of Bullet
  + One contains all the live enemies
  + One contains all the enemies that are inactive and need to be removed from the collection
  + One contains all the enemies that are hit and dead and need to be removed
  + One contains all the bullets the player fires
  + One contains all the bullets fired by the player that hit an enemy or flew too far without hitting anything
  + One contains all the bullets fired by the enemies
  + One contains all the bullets fired by the enemies that hit the player or travel too far from the viewport

## Content

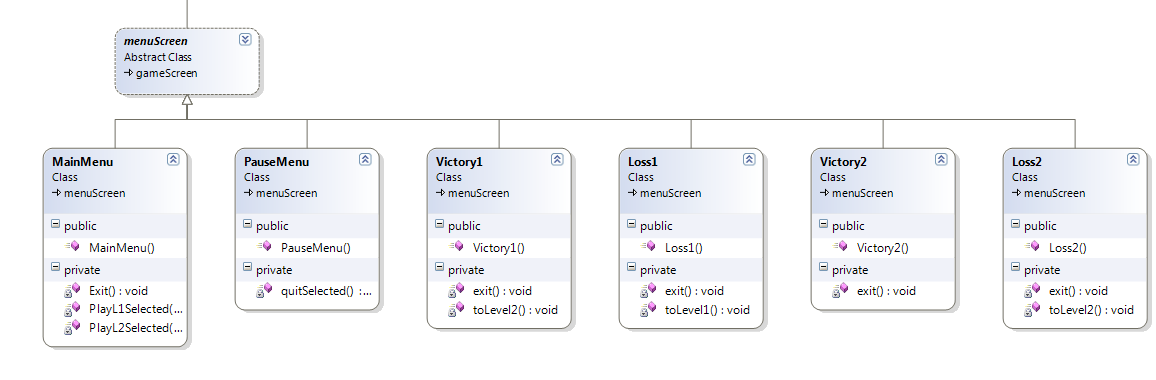
* Using XNA's contentmanager, loads in images, fonts, sounds
* Stored in the solution's content folder
  + The relevant images, sounds, and fonts are loaded when their level screen is called
  + The assets are not shared
  + Does not use a manager to deal with them once loaded in

## Physics, Statistics and AI

* Player movement is simple, with the A and D keys (or gamepad) increasing or decreasing the player's X screen position by 0.5f
  + If the X position becomes smaller than 0, warps player to the other side of the screen (viewport.Width - pSprite.Width)
  + Not able to move along the Y axis
* Rotation is handled using a float which is incremented or decremented around the value 0.0f
  + Rotation stops at 1.2 and -1.2, to give the illusion of a 'limited range of fire'
  + Speed is set to increment and decrement at 0.05f, very slow rotation
* Enemies are spawned at a random place on the X axis at the top of the screen, 30px in from either side
  + spawnTime = rand.Next(timeLeft \* 5, timeLeft \* 10);
  + Enemies spawn faster as their spawn timer is directly correlating to the time left timer
* Enemies have a list of bullets, which essentially mean the enemies share the same bullets
  + This means sometimes all enemies are firing 1 bullet, or only one is firing 5 bullets
* If an enemy's screenpos.X is smaller than yours, it will rotate to a clamped rotation angle, and vice versa
  + Persistent checks of position compared to player's
  + Does NOT follow the player perfectly, AI is not perfect
  + Enemy has a constant Y velocity of 1, they have no X velocity
* Bullets being fired take into account the rotation speed and velocity of the firing object
  + newBullet.velocity = new Vector2((float)Math.Sin(pSprite.getRotation()), (float)Math.Cos(pSprite.getRotation())) \* new Vector2(5f, -5f) + pSprite.getVel();
  + These Math methods get the rotational positions of the firing object and set the bullet's velocity to an appropriate movement, with swing and force pushing it faster if it is fired in the direction it is moving into
* Collision detection is basic bounding box
  + All sprites (enemies, player, and bullets) have a bounding box
  + Always checking for collisions between bullets and enemies, and enemies and player, and enemy bullets and player

## User Interface

* The navigation of the game is all selected by menu screens, which extend a main menu screen, which extends a game screen
* Player choices are recording from a KeyboardState or GamePadState which are sent to the relevant menu screens by the playerEvent class



### Flow Diagram for Menu Navigation

Main Menu

Level 1

Level 2

Pause

Level 1 Loss

Level 1 Win

Level 2 Win

Level 2 Loss

## Art and Video

* Content is loaded through XNA's contentmanager
* No engine used
* Images should be in .png or .bmp format

## Sounds and Music

* Loaded in through XNA's contentmanager
* No engine used
* Music should be in .mp3 format, sound effects in .wav format
* At least 128kbps
  + Music needs to be longer than 10 minutes to fit the game length
  + Sound effects to be SHORT, under 2 seconds
    - Repeating long sound effects do not sound good

## Dependency Graph

