SpacePig:
A game that has no pigs.

Nicholas Buquicchio • 12.06.2016

Introduction

SDL based game

- "bullet-hell" type game, player dodges waves of bullets

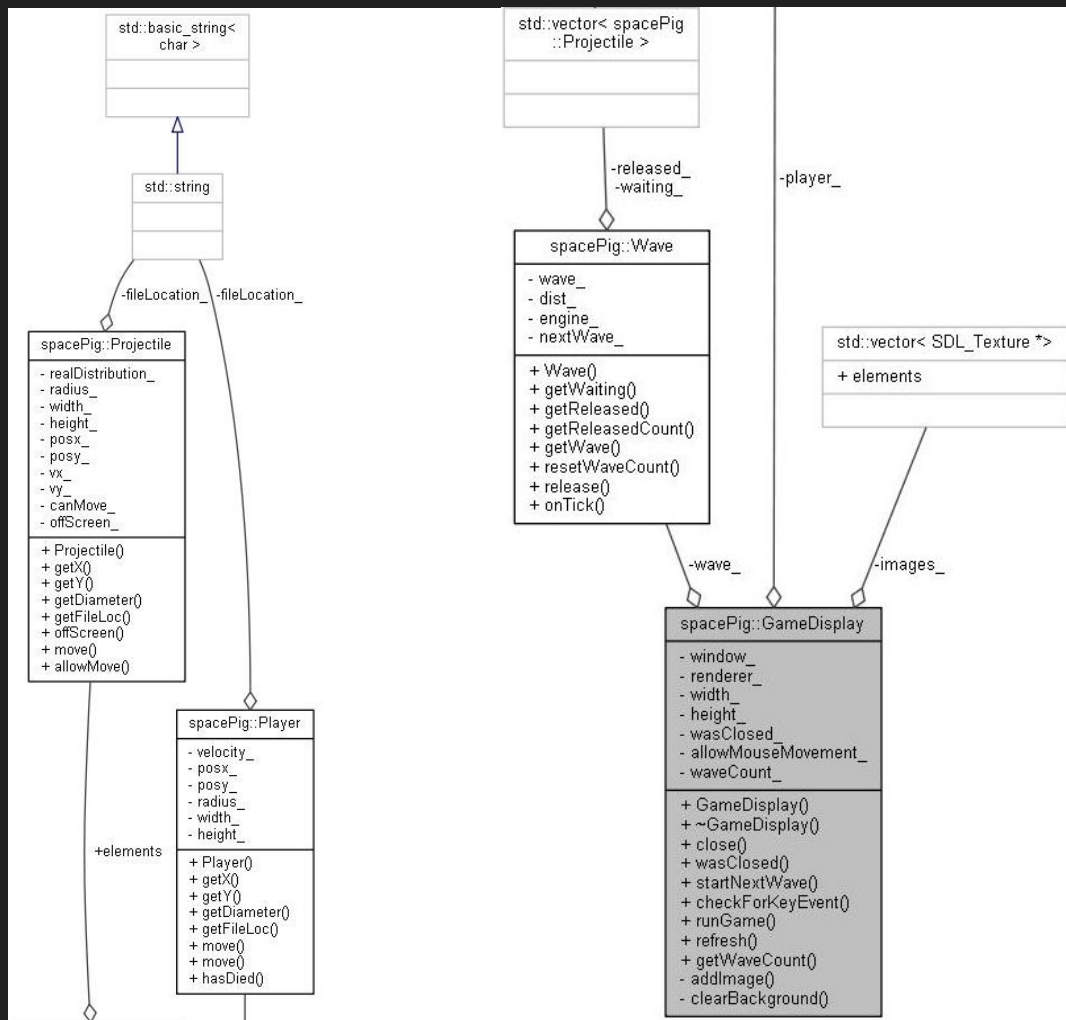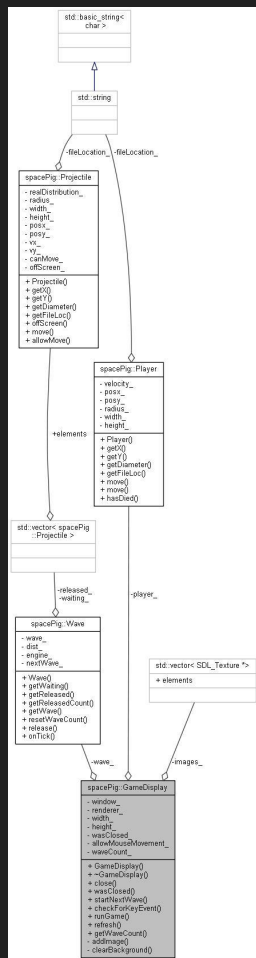- Uses key and mouse events for player movement

Wave system

- Each wave has projectiles - either waiting or released.

- Periodically releases projectiles for the player to dodge until none are left.

Use Case: Wave Based Bullet-hell

Step-by-step:

1. [System] The system loads and awaits user instruction

2. [Player] The player signifies to begin the game by hitting a key

3. [System] The system prepares the first wave

    - 4. [System] The system releases projectiles

5. [Player] The player uses keys/mouse to navigate the screen

    - 6. [System] The system interprets input into data, moving player where desired

7. [System] All projectiles for the wave have left the screen

    - 8. [System] Wait for a period before beginning the next wave

    - 4. [System] The system releases projectiles

9. [Player] The player collides with one of the projectiles

    - 10. [System] The system locks the player out of movement inputs

11. [System] The system awaits an exit or restart command

std::basic_string< char >

std::string

-fileLocation_ -fileLocation_

spacePig::Projectile
- realDistribution_
- radius_
- width_
- height_
- posx_
- posy_
- vx_
- vy_
- canMove_
- offScreen_

+ Projectile()
+ getX()
+ getY()
+ getDiameter()
+ getFileLoc()
+ offScreen()
+ move()
+ allowMove()

spacePig::Player
- velocity_
- posx_
- posy_
- radius_
- width_
- height_

+ Player()
+ getX()
+ getY()
+ getDiameter()
+ getFileLoc()
+ move()
+ move()
+ hasDied()

std::vector< spacePig ::Projectile >

+elements

-released_
-waiting_

spacePig::Wave
- wave_
- dist_
- engine_
- nextWave_

+ Wave()
+ getWaiting()
+ getReleased()
+ getReleasedCount()
+ getWave()
+ resetWaveCount()
+ release()
+ onTick()

std::vector< SDL_Texture *>
+ elements

-player_

-wave_   -images_

spacePig::GameDisplay
- window_
- renderer_
- width_
- height_
- wasClosed_
- allowMouseMovement_
- waveCount_

+ GameDisplay()
+ ~GameDisplay()
+ close()
+ wasClosed()
+ startNextWave()
+ checkForKeyEvent()
+ runGame()
+ refresh()
+ getWaveCount()
- addImage()
- clearBackground()

Projectile class

- Has an x,y coordinate and velocity
- Velocity is determined randomly, scaling with wave number
- Keeps track of whether it can move / is on screen

Wave class

- A collection of projectiles that are either waiting or released
- Waiting projectiles stay above the screen waiting to be fired off
- Released projectiles are allowed to move, and begin to shoot down from the top of the screen

+ Flaw: wave count as static field

Player class

- Has an x,y coordinate and velocity
- move() method is overloaded
    - One for keyboard movement
    - One for mouse movement

- When the player is hit, the game is over

GameDisplay class

- Built off of Display class, encapsulates SDL functionality
- runGame handles screen refreshing and gameplay
    - checks for key events
    - releases projectiles every .15 seconds until none left
    - stops game when the player has been hit

Conclusion

1. Was initially going to be a platformer with enemies, became too verbose

2. Hoped to implement a start screen / score system

Bibliography

Baclawski, Ken. "CS3520 Code Samples." *CS3520 Code Samples*. Northeastern University, n.d. Web. 04 Dec. 2016.

Foo, Mr. "Lazy Foo' Productions." *Beginning Game Programming V2.0*. N.p., n.d. Web. 04 Dec. 2016.

Nick Dahill for all art purposes