statePtr == std::shared_pointer<State>
dynamicObjPtr ==
std::shared_pointer<DynamicObject>
staticObjPtr == std::shared_ptr<StaticObject>

Namespace Collision

+ PixelPerfectTest(sf::Sprite, sf::Sprite, sf::Uint8): bool
+ getPixelsOverlap(sf::Sprite, sf::Sprite, sf::Uint8): std::vector<sf::vector2f>

Constants

+ textureExplosion: sf::Texture
+ textureIceBlock: sf::Texture
+ textureShip: sf::Texture
+ textureBomb: sf::Texture
+ textureProjectile: sf::Texture
+ textureBombSite: sf::Texture
+ font: sf::Font
+ tickRate: long
- levelTextures: std::vector<sf::Texture>
- bombSites: std::vector<std::vector<sf::Vector2f>>
- playerLocations: std::vector<std::vector<sf::Vector2f>>
- levelTimers: std::vector<int>

+ instance: static Constants

+ Constants()

+ getLevelTexture(int level): sf::Texture&
+ getLevelBombSites(int level): std::vector<sf::Vector2f>&
+ getPlayerLocation(int level): std::vector<sf::Vector2f>&
+ getLevelTimer(int level): int

- loadLevels(): void
- loadTexture(std::string const& address): void
- initializeTextures(): void

State

keys;: std::map<sf::KeyBoard::Key, bool>
currentLevel: int

+ State(int)
+ ~State()

+ onKeyPressed(sf::KeyBoard::Key): void
+ onKeyReleased(sf::KeyBoard::Key): void
+ updateState(sf::View, bool): StatePtr virtual = 0
+ renderState(sf::RenderWindow): void virtual = 0
+ getKeys(): std::map<sf::KeyBoard::Key, bool>

Game State

- endTimer: sf::Clock
- endTimerStarted: bool
- endTimerText: sf::Text
- shipDied: bool
- bombsPlaced: int
- totalSites: int
- currentEndTime: int
- dynamicObjects: std::vector<dynamicObjPtr>
- dynamicAddList: std::vector<dynamicObjPtr>
- staticObjects: std::vector<staticObject>
- staticAddList: std::vector<staticObjPtr>

+ GameState(int, sf::View)

+ createObject(int, sf::Vector2f): void
+ addObject(dynamicObjPtr): void
+ addObject(staticObjPtr): void
+ removeObjects(): void
+ updateState(sf::View, bool): StatePtr override
+ renderState(sf::RenderWIndow): void override
+ setShipDied(): void
+ checkTimeOut: bool
+ addOneBombPlaced(): void
+ getCometSize(): std::pair<float, float>
+ getEndTimerText(): sf::Text
+ adjustView(sf::View): void override

- initializeKeys(): void
- initializeText(sf::View, std::string): void
- initializeIceBlocks(): void
- startUpCollisions(dynamicObjPtr): bool
- initializeObjects(): void
- initializeGame(): void
- updateObjectLIsts(): void
- updateExplosion(): void
- updateDynamicObjects(sf::View): void
- collisionsWithStatic(sf::View): void
- collisionsWithDynamic(sf::View): void
- getShip(): dynamicObjPtr
- shipLeftComet(): bool
- hasWon(): bool
- gameOver(): bool

1

MenuState

header: sf::Text
enterText: sf::Text
escapeText: sf::Text

+ MenuState(int, sf::View, std::string)

+ updateState(sf::View, bool): statePtr virtual override
+ renderState(sf::RenderWIndow): void override

- adjustView(sf::View): void
- initializeKeys(): void
- initializeText(sf::View, std::string): void

PausMenu

- currentGameState: statePtr

+ PausMenuState(int, sf::View, statePtr)

+ updateState(sf::View, bool): statePtr override

WinState

- finalWin: bool

+ WinState(int, sf::View)

+ updateState(sf::View, bool): statePtr override

- initializeTextLocal(sf::View, std::string): void

Object

\# sprite: sf::Sprite
\# alive: bool

+ Object(sf::Vector2f)
+~Object(): virtual
+ isAlive(): bool
+ getSprite(): sf:Sprite
+ draw(sf::RenderWindow): void

3..*

DynamicObject

\# mass: float
\# direction: sf::vector2f
\# orientation: sf::vector2f
\# speed: float
\# radius: float

+ DynamicObject(sf::Vector2f, float,
sf::Vector2f, float, float)
+ ~DynamicObject()

+ update(sf::View, GameState): virtual void
+ collisionUpdate(GameState): virtual void = 0
+ elasticCollision(dynamicObjPtr, sf::View, GameState): void
+ elasticCollision(staticObjPtr, sf::View, GameState): void
+ inelasticCollision(dynamicObjPtr): dynamicObjPtr
+ getSpeed(): float
+ setAliveStatus(bool): void

\# rotateVector(sf::Vector2f , float): sf::Vector2f
\# normalizeSpeedDirection(sf::Vector2f): void
\# calculateNormalizedVector(sf::Vector2f): sf::Vector2f

- getAverageCollidingPixel(sf::Sprite): sf::Vector2f
- adjustPosition(sf::VIew, sf::Vector): void
- calculateVector2fLength(sf::Vector2f): float

StaticObject

+ StaticObject(sf::Vector2f)
+~StaticObject(): virtual

Explosion

- clock: sf::Clock

+ Explosion(sf::Vector2f)
+ update(): void

Comet

+ Comet(int)

BombSite

- bombPlaced: bool

+ BombSite(sf::Vector2f)
+ collisionUpdate(GameState): void

Bomb

- clock: sf::Clock
- timerSec: int
- safe: bool

+ Bomb(sf::Vector2f, sf::Vector2f, float)

+ update(sf::View, GameState): void
+ collisionUpdate(GameState): void

- initializeTexture(): void
- setOrigin(): void
- setRadius(): void

Ship

- hp: float
- maxHp: float
- shieldHp: float
- shieldHp: float
- maxShieldHp: float
- bombs: int
- shipAcceleration: float
- shipRotationSpeed: float
- hpCountdown: sf::Text
- shieldHpCountdown: sf::Text
- bombsLeft: sf::Text

+ Ship(sf::Vector2f)
+ update(sf::View, GameState): void override
+ collisionUpdate(GameState): void override
+ drawText(sf::RenderWindow): void

- initializeText(): void
- initializeTexture(): void
- setRadius(): void
- setOrigin(): void
- updateView(sf::View, GameState): void
- updateTextContent(): void
- updateTextPositions(sf::View , GameState): void
- updateSpeedDirection(GameState): void
- updateRotation(sf::View, GameState): void
- updateWeapons(GameState): void
- regenerateShield(): void

Projectile

_____

+ Projectile(sf::Vector2f, sf::Vector2f, float)
+ collisionUpdate(GameState): void

- initializeTexture(): void
- setOrigin(): void
- setRadius(): void

IceBlock

- sf::Clock: clock
- blockSize: int
- hp: int
- age: int

+ IceBlock(sf::Vector2f)
+ IceBlock(sf::Vector2f, float, sf::Vector2f)

+ update(sf::View, GameState): void
+ collisionUpdate(GameState): void
+ getAge(): int

- initializeTexture(): void
- setRadius(): void
- setOrientation(): void
- setOrigin(): void
- splitIceBlock(GameState): void