COSC345 Program
@author; Max Stewart //1086706, Elbert Alcantara //4435223, Sarang Han //5098495

Developer Documentation

Please find in this document descriptions of what each method will do, this will help developers in the future to maintain and extend our program.

COSC345 Program
@author; Max Stewart //1086706, Elbert Alcantara //4435223, Sarang Han //5098495

Changes since the Beta release

GameStates.cpp
Implemented to shorten code; code pulled from HardState.cpp & EasyState.cpp (which existed in alpha version).
HardState was removed altogether containing only the original EasyState.cpp, which is the main and only state to shorten code.
Music was added directly in class instead of a loop executed at game state. Life points was added. Textures were also changed.

Methods:
GamesStates::GameState(GameDataRef data): _data(data)
@param GameDataRef data: data necessary for both easy state and game state
Checks which game state is chosen by the player.
Loads and sets textures needed for both easy and hard game states.

void GameStates::Setup(int x)
@param int x: position of player
Sets hasChosen = true so player can be drawn && sets players position and ensures tile type (e.g. Bomb, Life) are allocated after player clicks starting position or else they may click a bomb.
Calculates how many mines are beside each tile to _grid_under array if tile is not a bomb.

void GameStates::HandleInput()
This method handles player movement. Ensures player stays within bounds of the game board, and reveals the type of tile it is located in and calls functions accordingly.

void GameStates::Update(float dt)
@param float dt: interpolation time
Moves player position by calling Move function in Player class and takes 6 * interpolation time seconds to move.

void GameStates::Draw(float dt)
@param float dt: interpolation time in seconds
GameState background and sprites are drawn here.

void GameStates::Win()
Makes private bool hasWon true which is required move to the GameWon mode in SplashState.
Executes _player->MoveOff(int x) which makes the player model walk down and off the screen.

COSC345 Program
@author; Max Stewart //1086706, Elbert Alcantara //4435223, Sarang Han //5098495

Player.cpp
Player::Implemented to add player model in game.
Added method MoveOff since Beta.

Methods:
Player::Player(GameDataRef data) : _data
@param GameDataRef data ; data necessary for player movement
This method sets and loads texture necessary for in game player sprite, it initializes variables needed for methods later on.

bool Player::PlayerMoving()
@return true: if player is moving

bool Player::PlayerChosen()
@return true: if player has picked a start position

void Player::setPos(int x)
@param int x: where player chooses to start in x axis within bounds of game board with y always within top row of game board
This method sets players initial position, and triggers initialization of tile under type.

sf::Vector2i Player::GetPos()...
@return sf::Vector2i(xPosition, yPosition): returns play position

void Player::Explode(int newX, int newY)
@param int newX: position of where "explosion" needs to be set in the x axis
@param int newY: position of " explosion" needs to be set in the y axis
Sets position of explosion sprite to where is needed.

void Player::MovePlayer(int newX, int newY, int spriteShown)
@param int newX: picked next position of player ; players next choice
@param int newY: picked next position of player; players next choice
@param int spriteShown: where the sprites shown
This method accounts for out of bounds exceptions - checks if next move is valid or not.

void Player::Draw()
Draws in game character

void Player::Move(float dt)
@param float dt: vector difference of the movement of player
This method handles player movement//change.

void Player::MoveOff(int x)
@param int x: current position in x of player
Makes player moved down off the screen.

COSC345 Program
@author; Max Stewart //1086706, Elbert Alcantara //4435223, Sarang Han //5098495

Main.cpp
Creates an instance of Game
Has the width and height of what is defined in definitions.hpp and window is called 'beta'

MainMenuState.cpp
Play button is swapped out to Save-Daughter and Instruction button. Button Textures were changed. Music added directly in the class instead of the a loop executed at game state.

Methods:
MainMenuState::MainMenuState(GameDataRef) : _data(data)
@param GameDataRef data: Contains needed data of game
This method loads a new reference of _data which is concordant with the data MainMenuState needs. It also loads and sets textures needed for MainMenuState.
It also loads the background & 'Play' icon and sets their positions.
It also sets and gets private variables declared in the coinciding .hpp file.

void MainMenuState::HandleInput()
Closes window if window is requested to be closed (no data present).
If sprite (the 'play' icon) is pressed, it takes player to game screen from the main menu.

Void MainMenuState::Update(float dt)
@param float dt: interpolation time to update state
No data is updated in this method of this state.

Void MainMenuState::Draw(float dt)
@param float dt: current time, not used in this method in this state
Loaded _data gets cleared in window & draws updated data to window.

HowToState.cpp
This class was implemented to add a "how to play" screen

Definitions.hpp
Size has been enlarged to increase challenge, and a few images has been changed. Implemented to make such changes easier.
Contains game constants such as game board width, texture folder location.

AssetManager.cpp
No Changes.

Methods:
Void AssetManager::LoadTexture(std::string name, std::string filename)
@param std::string name: assigns a string to represent file name that is defined in the definitions.cpp file
@param std::string fileName: file name that defines pathway of file thats loaded/set

Sets texture to private variables - needed for encapsulation.

sf::Texture &AssetManager::GetTexture(std::string name)
@param std::string name: string that is assigned to represent file name that is defined in the definitions.cpp file
Gets texture set to private variables- needed for encapsulation.

void LoadFont(std::string name, std::string fileName)
@param std::string name: assigns a string to represent file name that is defined in the definitions.cpp file
@param std::string fileName ; file name that defines pathway of file thats loaded/set
Sets font to private variables.

sf::Font &AssetManager::GetFont(std::string name)
@param std string name: string that is assignment to represent file name that is defined in the definitions.cpp file
Gets Font set to private variables- needed for encapsulation.

InputManager.cpp
Not Changed since Beta.

Methods:
bool InputManager::IsSpriteClicked(sf::Sprite object, sf::Mouse::Button button, sf::RenderWindow &window)
@param sf::Sprite object: sprite (icon) to be clicked
@param sf::Mouse::Button button: any mouse click operation
@param sf::RenderWindow &window: window that input manager is currently managing
@return bool IsSpriteClicked: returns true if sprite clicked
This method finds the boundaries of the sprite and if a mouse button is clicked within those boundaries, returns true, if not, false

sf::Vector2i InputManager::GetMousePosition(sf::RenderWindow &window)
@param sf::RenderWindow &window: window that input manager is currently working on
@return GetMousePosition: returns position of mouse
This method returns the position of mouse after mouse click has occurred.

Game.cpp
No Changes since Beta

Methods:
Game::Game(int width, int height, std::string title)
@param int width ; width of game window
@param int height ; height of game window
@param std::string title ; title of game
Creates the games window and takes player to splash screen.

Void Game::Run()
This method uses time to update and handle inputs between different frames && finds all timing values needed to run game.

COSC345 Program
@author; Max Stewart //1086706, Elbert Alcantara //4435223, Sarang Han //5098495

SplashState.cpp
We added a few photos in loading the game to set up a back story.
Also added introduction narration for story. Class handles the introduction screen, Win, and Lose screen.

Methods:
SplashState::SplashState(GameDataRef data) : _data(data)
@param GameDataRef data: Contains needed data of game
This method Loads a new instance of state, splashstate for splash screen && Loads and sets textures needed for splash screen; also sets position of textures

Void Splashstate::HandleInput()
Closes window if window is requested to be closed (nothing in _data)

Void SplashState::Update(float dt)
@param float dt: interpolation times to update state
This method takes player to main menu screen after 9.8 (in definitions folder SPLASH_TITLE_SHOW_TIME is assigned to a float of 9.8) seconds.

Void SplashState::Draw(float dt)
@param float dt: current time, used to know which object to draw at a certain time
This method clears and displays window && Draws window and what was loaded and set from SplashState::SplashState(GameDataRef data) : _data(data) according to a certain period of time.

StateMachine.cpp
No Changes since Beta. Used to change Game States.

Methods:
void StateMachine::AddState(StateRef newState, bool isReplacing)
@param StateRef newState ; new state that has replaced old state
@param bool isReplacing ; returns true if new state is created / is in line for 'stack' if not, false
Sets variables needed to help
ProcessStateChanges()

void StateMachine::ProcessStateChanges()
If statements which check what is in front of 'line' (or stack) check if current state has been replaced or if a new state is 'in line' to be created, if a new state is in 'line' to be created, current state is 'popped' off the line and the ascending one is created/ replaces current state

StateRef &StateMachine::GetActiveState()
@returns current active state

We have used SFML (3rd party library) which needs to be installed correctly for our program to run

COSC345 Program
@author; Max Stewart //1086706, Elbert Alcantara //4435223, Sarang Han //5098495

The files required to install SFML is already present in our GitHub repositories under extLib in the project folder.

SFML can also be downloaded from:
https://www.sfml-dev.org/download/sfml/2.5.1/
Once downloaded all SFML files need to be moved into the project root foley

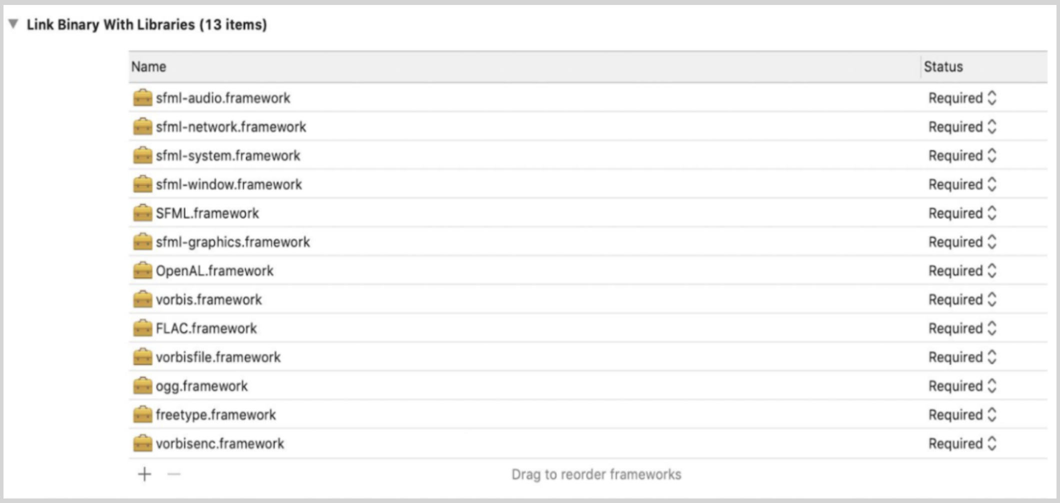
For Xcode build phases & build search paths need to be configured for SFML to work. Build phases>>Link binary with libraries >> all the .framework files in both extlib && framework folder (should be 13 in total)

Under header search paths, the include file in the SFML file needs to be added

Under compiled sources (still in build phase) all folders should be present, so it should look like this:

Link for SFML setup:
Https://www.sfml-dev.org/tutorials/2.5/start-osx.php

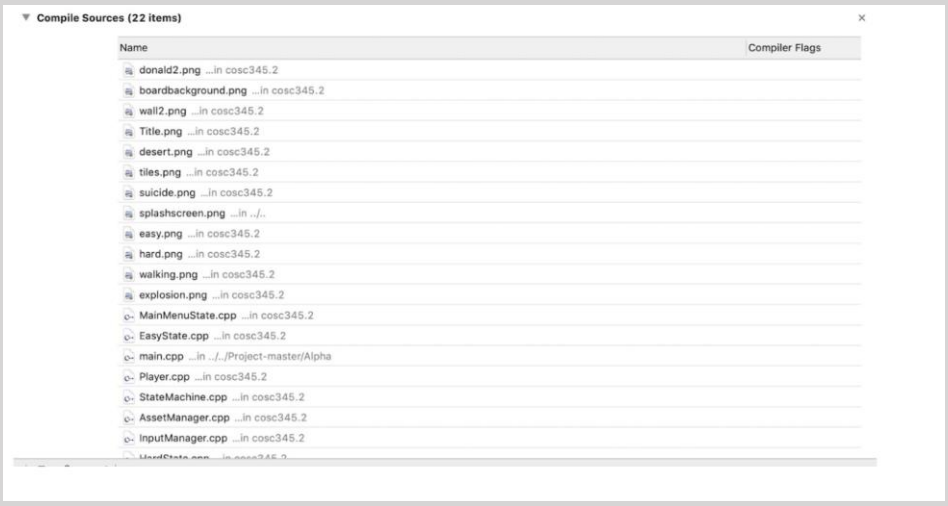
COSC345 Program
@author; Max Stewart //1086706, Elbert Alcantara //4435223, Sarang Han //5098495



▼ Link Binary With Libraries (13 items)

Name	Status
sfml-audio.framework	Required
sfml-network.framework	Required
sfml-system.framework	Required
sfml-window.framework	Required
SFML.framework	Required
sfml-graphics.framework	Required
OpenAL.framework	Required
vorbis.framework	Required
FLAC.framework	Required
vorbisfile.framework	Required
ogg.framework	Required
freetype.framework	Required
vorbisenc.framework	Required

Drag to reorder frameworks



▼ Compile Sources (22 items)

Name	Compiler Flags
donald2.png ...in cosc345.2	
boardbackground.png ...in cosc345.2	
wall2.png ...in cosc345.2	
Title.png ...in cosc345.2	
desert.png ...in cosc345.2	
tiles.png ...in cosc345.2	
suicide.png ...in cosc345.2	
splashscreen.png ...in ../..	
easy.png ...in cosc345.2	
hard.png ...in cosc345.2	
walking.png ...in cosc345.2	
explosion.png ...in cosc345.2	
MainMenuState.cpp ...in cosc345.2	
EasyState.cpp ...in cosc345.2	
main.cpp ...in ../../Project-master/Alpha	
Player.cpp ...in cosc345.2	
StateMachine.cpp ...in cosc345.2	
AssetManager.cpp ...in cosc345.2	
InputManager.cpp ...in cosc345.2	