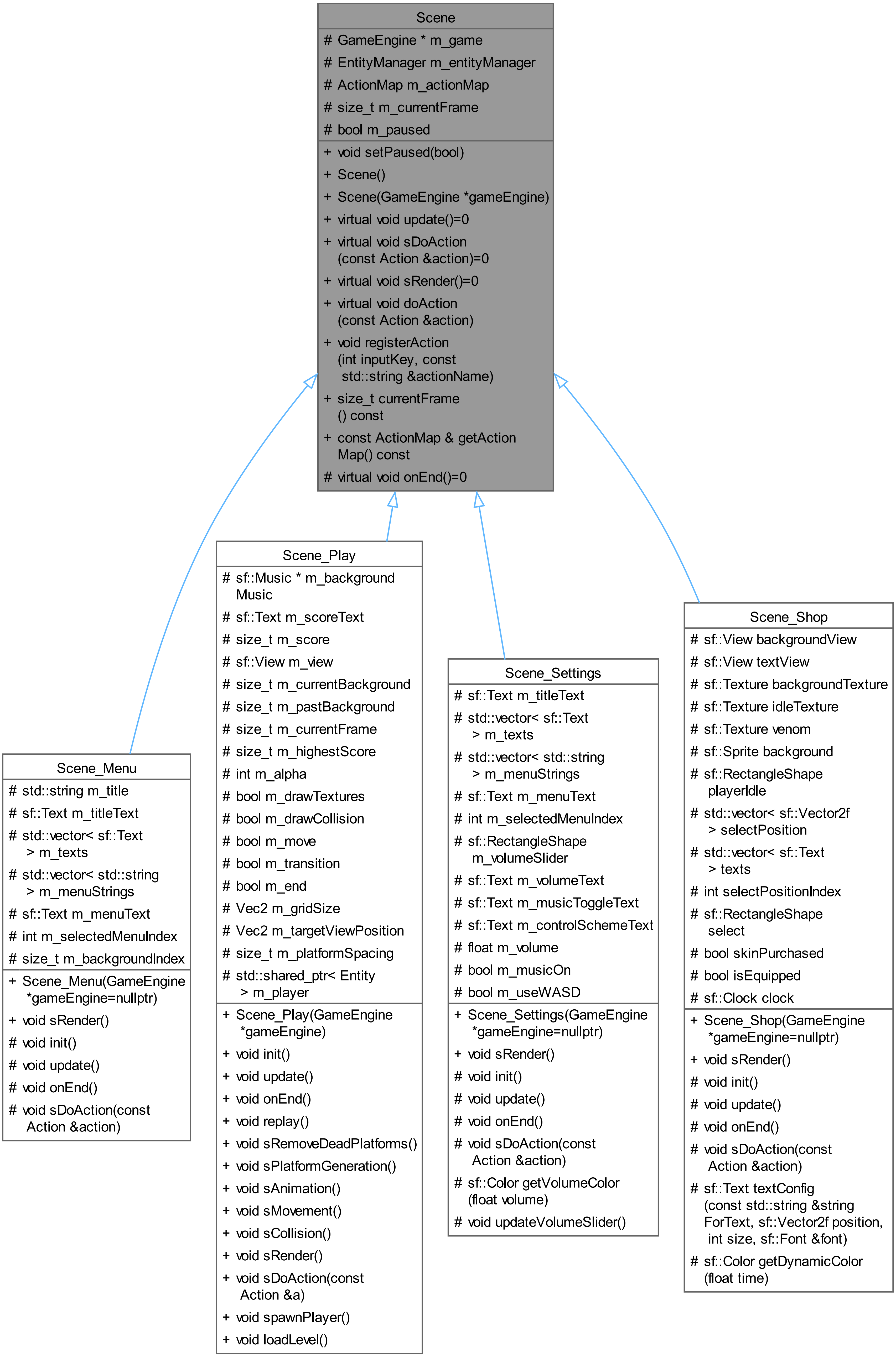
Scene
# GameEngine * m_game
# EntityManager m_entityManager
# ActionMap m_actionMap
# size_t m_currentFrame
# bool m_paused
+ void setPaused(bool)
+ Scene()
+ Scene(GameEngine *gameEngine)
+ virtual void update()=0
+ virtual void sDoAction (const Action &action)=0
+ virtual void sRender()=0
+ virtual void doAction (const Action &action)
+ void registerAction (int inputKey, const std::string &actionName)
+ size_t currentFrame () const
+ const ActionMap & getActionMap() const
# virtual void onEnd()=0
Scene_Menu
# std::string m_title
# sf::Text m_titleText
# std::vector< sf::Text > m_texts
# std::vector< std::string > m_menuStrings
# sf::Text m_menuText
# int m_selectedMenuIndex
# size_t m_backgroundIndex
+ Scene_Menu(GameEngine *gameEngine=nullptr)
+ void sRender()
# void init()
# void update()
# void onEnd()
# void sDoAction(const Action &action)
Scene_Play
# sf::Music * m_backgroundMusic
# sf::Text m_scoreText
# size_t m_score
# sf::View m_view
# size_t m_currentBackground
# size_t m_pastBackground
# size_t m_currentFrame
# size_t m_highestScore
# int m_alpha
# bool m_drawTextures
# bool m_drawCollision
# bool m_move
# bool m_transition
# bool m_end
# Vec2 m_gridSize
# Vec2 m_targetViewPosition
# size_t m_platformSpacing
# std::shared_ptr< Entity > m_player
+ Scene_Play(GameEngine *gameEngine)
+ void init()
+ void update()
+ void onEnd()
+ void replay()
+ void sRemoveDeadPlatforms()
+ void sPlatformGeneration()
+ void sAnimation()
+ void sMovement()
+ void sCollision()
+ void sRender()
+ void sDoAction(const Action &a)
+ void spawnPlayer()
+ void loadLevel()
Scene_Settings
# sf::Text m_titleText
# std::vector< sf::Text > m_texts
# std::vector< std::string > m_menuStrings
# sf::Text m_menuText
# int m_selectedMenuIndex
# sf::RectangleShape m_volumeSlider
# sf::Text m_volumeText
# sf::Text m_musicToggleText
# sf::Text m_controlSchemeText
# float m_volume
# bool m_musicOn
# bool m_useWASD
+ Scene_Settings(GameEngine *gameEngine=nullptr)
+ void sRender()
# void init()
# void update()
# void onEnd()
# void sDoAction(const Action &action)
# sf::Color getVolumeColor (float volume)
# void updateVolumeSlider()
Scene_Shop
# sf::View backgroundView
# sf::View textView
# sf::Texture backgroundTexture
# sf::Texture idleTexture
# sf::Texture venom
# sf::Sprite background
# sf::RectangleShape playerIdle
# std::vector< sf::Vector2f > selectPosition
# std::vector< sf::Text > texts
# int selectPositionIndex
# sf::RectangleShape select
# bool skinPurchased
# bool isEquipped
# sf::Clock clock
+ Scene_Shop(GameEngine *gameEngine=nullptr)
+ void sRender()
# void init()
# void update()
# void onEnd()
# void sDoAction(const Action &action)
# sf::Text textConfig (const std::string &stringForText, sf::Vector2f position, int size, sf::Font &font)
# sf::Color getDynamicColor (float time)