IData.hpp : INTERFACE

Methode :

- Enum class DataType :
  - Contient l'ensemble des formes et autres objets disponible.
- Virtual ~IData () :
  - Destructeur par défaut.
- Virtual DataType const &getShape() const :
  - Récupère la forme (DataType) de l'objet.
- virtual void setShape(DataType const &shape) :
  - Permet de modifier le DataType de l'objet.
- virtual Position const &getPosition() const
  - Permet de récupérer la position de l'objet.
- virtual void setPosition(Position const &position)
  - Permet de modifier la position de l'objet.
- virtual void reset(void)
  - Permet de reset.

AData.hpp : IData

Methode :

- Virtual ~AData () :
  - Destructeur par défaut.

- DataType const &getShape() const :
  - Récupère la forme (DataType) de l'objet en fonction du DataType

- virtual void setShape(DataType const &shape) :
  - Permet de modifier le DataType de l'objet.

- virtual Position const &getPosition() const
  - Permet de récupérer la position de l'objet.

- virtual void setPosition(Position const &position)
  - Permet de modifier la position de l'objet.

- virtual void setPositionX(double const x);
  - Permet de modifier la position X.

- virtual void setPositionY(double const y);
  - Permet de modifier la positon Y.

- virtual void setPositionZ(double const z);
  - Permet de modifier la position Z (3D).

- virtual void reset(void)
  - Permet de reset.

Protected:

- DataType _shape
  - Contient le type de l'objet.
- Position _position
  - Contient la position de l'objet.

AScene.hpp : AData

Methode :

- Float getIncX() const
  - Permet de récupérer la position X (3D).
- virtual void setIncX(float incX);
  - Permet de modifier la position X (3D).
- Float getIncY() const
  - Permet de récupérer la position Y (3D).
- virtual void setIncY(float incY);
  - Permet de modifier la position Y (3D).
- virtual void reset(void)
  - Permet de reset.

Protected:

- Float _incX
  - Contient la position X (3D).
- Float _incY
  - Contient la position Y (3D).

AScene.hpp->AData (Permet de récupérer les positons X, Y, et Z d'un objet pour un visionnage 3D, stock sa position X, Y et Z

Audio.hpp : AData

Methode :

- explicit Audio (std ::string const & audio, unsigned int intensity = 100, bool repeat = false)
    - Constructeur par défaut.

- ~Audio ()
    - Destructeur par défaut.

- std::string const &getAudio() const
    - Permet de récupérer le fichier Audio.

- bool getRepeat() const
    - Permet de récupérer la valeur du booléen de répétition.

- void setRepeat(bool repeat)
    - Permet de modifier la valeur du booléen de répétition.

- unsigned int getIntensity() const
    - Permet de récupérer la valeur d'intensité.

- void setIntensity(unsigned int intensity)
    - Permet de modifier la valeur d'intensité.

- void reset(void)
    - Permet de reset.

Protected:

- Std::string _audio
    - Contient le nom du fichier audio.
- Unsigned int _intensity
    - Contient la valeur d'intensité.
- Bool _repeat
    - Contient la valeur du booléen de répétition.

Audio.hpp->AData (Permet de charger un fichier audio et de gérer son intensité ainsi que la répétition ou non du morceau, stock le nom du morceau, son intensité ainsi qu'un booléen de répétition)

AVisual.hpp : AData

Methode :

- explicit Audio (std ::string const & audio, unsigned int intensity = 100, bool repeat = false)
    - Constructeur par défaut.

- ~AVisual ()
    - Destructeur par défaut.

- std::string const &getTexture() const
    - Permet de récupérer la texture.

- virtual void setTexture(Texture const &texture)
    - Permet de modifier la texture.

- unsigned int getZIndex() const
    - Permet de récupérer la valeur Z (3D) .

- virtual void setZIndex(unsigned int zIndex)
    - Permet de modifier la valeur Z (3D)..

- unsigned int getId() const
    - Permet de récupérer l'ID de l'objet.

- virtual void setId(unsigned int id)
    - Permet de modifier l'ID de l'objet.

- void reset(void)
    - Permet de reset.

Protected:

- Texture _texture
    - Contient la texture de l'objet.
- Unsigned int _zIndex
    - Contient la valeur de Z (3D).
- Unsigned int _id
    - Contient l'ID de l'objet.

AVisual.hpp->AData (Permet de gérer les objets pour ajouter des textures, obtenir leurs positions ainsi que leurs ID, stock la texture, la positon X,Y, Z (3D) et l'ID)

Camera.hpp : AScene

Methode :

- explicit Camera (Position const &pos, float incX = 0, float incY = 0)
    - Constructeur par défaut.

- ~Camera ()
    - Destructeur par défaut.

- void reset(void)
    - Permet de reset.

Camera.hpp->AScene (Place une caméra pour une vision 3D en fonction des position X, Y et Z récupérer grâce à la class AScene)

Cube.hpp : AVisual

Methode :

- explicit Cube(Position const &pos, Position const &size, Texture const &text = Texture(), unsigned int zIndex = 0, unsigned int id = 0)
    - Constructeur par défaut.

- ~Cube ()
    - Destructeur par défaut.

- Position const &getSize() const;
    - Permet de récupérer la taille de l'objet.

- void setSize(Position const &size)
    - Permet de modifier la taille de l'objet.

- bool inLine(double a, double new_a, double size) const.
    - Permet de verifier les hitbox.

- void reset(void)
    - Permet de reset.

Protected:

- Position _size
    - Contient la taille de l'objet.

Cube.hpp->AVisual (Crée un cube à une position, une taille et une Texture choisie, stock sa position)

Light.hpp : AVisual

Methode :

- explicit Light(Position const &pos, float incX = 0, float incY = 0)
    - Constructeur par défaut.

- ~Light ()
    - Destructeur par défaut.

- void reset(void)
    - Permet de reset.

Light.hpp->AScene (Permet de positionner une lumière dans une scène 3D à la position X, Y et Z)

Sphere.hpp : AData

Methode :

- explicit Sphere(Position const &pos, float radius, Texture const &text = Texture(), unsigned int zIndex = 0)
    - Constructeur par défaut.

- ~Sphere ()
    - Destructeur par défaut.

- float getRadius() const
    - Permet de récupérer le radiant de l'objet.

- void setRadius(float radius)
    - Permet de modifier la radiant de l'objet.

- void reset(void)
    - Permet de reset.

Protected:

- float _radius
    - Contient la valeur du radiant de l'objet.

Sphere.hpp->AVisual (Crée une sphère à une position, un radiant, une texture et les positions X, Y et Z choisies, stock son radiant)

Text.hpp : AVisual

Methode :

- enum Align
  - Contient les différentes possibilitées d'alignement du texte.

- explicit Text (std::string const &text, unsigned int size, Position const &pos = Position(), unsigned int zIndex = 0)
  - Constructeur par défaut.

- ~Text ()
  - Destructeur par défaut.

- std::string const &getText() const
  - Permet de récupérer le texte.

- void setText(std::string const &text)
  - Permet de modifier le texte.

- Align const &getAlign() const
  - Permet de récupérer la méthode d'alignement du texte.

- void setAlign(Align const &align)
  - Permet de modifier la valeur d'alignement du texte.

- unsigned int getSize() const
  - Permet de récupérer la taille du texte

- void setSize(unsigned int size)
  - Permet de modifier la taille du texte.

- void reset(void)
  - Permet de reset.

Protected:

- Std::string _text
  - Contient le texte.
- Align _align
  - Contient la valeur d'alignement.

- Unsigned int _size
    - Contient la taille du texte.

Text.hpp->AVisual (Permet d'afficher du texte à l'écran avec une taille et une position ainsi qu'une positon Z choisis. Stock le texte à afficher, le mode d'alignement ainsi que la taille)

Color.hpp

Methode :

- Color (unsigned char const r = 0, unsigned char const g = 0, unsigned char const b = 0, unsigned char const a = 0)
    - Contient les différentes possibilitées d'alignement du texte.

- ~Color ()
    - Destructeur par défaut.

- unsigned char getR() const
    - Permet de récupérer la valeur R du code couleur RGBA.

- unsigned char getG() const
    - Permet de récupérer la valeur G du code couleur RGBA.

- unsigned char getB() const
    - Permet de récupérer la valeur B du code couleur RGBA.

- unsigned char getA() const
    - Permet de récupérer la valeur A du code couleur RGBA.

- void setR(unsigned char const r);
    - Permet de changer la valeur R du code couleur RGBA.

- void setR(unsigned char const g);
    - Permet de changer la valeur G du code couleur RGBA.

- void setR(unsigned char const b);
    - Permet de changer la valeur B du code couleur RGBA.

- void setR(unsigned char const A);
    - Permet de changer la valeur A du code couleur RGBA.

- void setRGBA(unsigned char const x, unsigned char const y, unsigned char const z, unsigned char const a);
    - Permet de changer la valeur R, G, B, A du code couleur RGBA.
    -

Private:

- Unsigned char _r
  - Contient la valeur R du code couleur RGBA
- Unsigned char _g
  - Contient la valeur G du code couleur RGBA.
- Unsigned char _b
  - Contient la valeur B du code couleur RGBA.
- Unsigned char _a
  - Contient la valeur A du code couleur RGBA.

Color.hpp (Prend en paramètre un code couleur R, G, B,A)

Model3D.hpp

Methode :

- Model3D (std::string const &object = "", Position const &position = Position(), Position const &_scale = Position(1, 1, 1), Position const &_rotation = Position())
    - Constructeur par défaut.

- ~Model3D ()
    - Destructeur par défaut.

- void setObject(std::string const &object)
    - Set le type d'un Objet.

- std::string const &getObject(void) const
    - Récupère le type de l'objet.

- void setScale(Position const &scale)
    - Permet modifier la taille de l'objet 3D.

- void setScaleX(double x)
    - Permet de modifier la taille X de l'objet 3D.

- void setScaleY(double y)
    - Permet de modifier la taille Y de l'objet 3D.

- void setScaleZ(double z)
    - Permet de modifier la taille Z de l'objet 3D.

- Position const &getScale(void) const
    - Permet de récupérer la taille de l'objet 3D.

- void setPosition(Position const &position)
    - Permet de modifier la position de l'objet 3D.

- void setPositionX(double x)
    - Permet de modifier la position X de l'objet 3D.

- void setPositionY(double y)
    - Permet de modifier la position Y de l'objet 3D.

- void setPositionZ(double z)
    - Permet de modifier la position Z de l'objet 3D.

- Position const &getPosition(void) const
    - Permet récupérer la position de l'objet sur la position XYZ de l'objet 3D.

- void setRotation(Position const &rotation)
    - Permet de mettre en place la rotation d'un objet.

- void setRotationX(double x)
    - Permet de configurer la rotation sur la position X de l'objet.

- void setRotationY(double y)
    - Permet de configurer la rotation sur la position Y de l'objet.

- void setRotationZ(double z)
    - Permet de configurer la rotation sur la position Z de l'objet.

- Position const &getRotation(void) const
    - Permet de récupérer les positions XYZ de la rotation de l'objet.

Private:

- std::string _object
    - Chemin du fichier 3D.

- Position _scale;
    - Contient la taille de l'objet.

- Position _position;
    - Contient la position de l'objet.

- Position _rotation;
    - Contient la position de la rotation de l'objet.

Position.hpp

Methode :

- Position (double const x = 0, double const y = 0, double const z = 0)
  - Constructeur par défaut.

- ~Position ()
  - Destructeur par défaut.

- double getX() const
  - Permet de récupérer la position X de l'objet.

- double getY() const
  - Permet de récupérer la position Y de l'objet.

- double getZ() const
  - Permet de récupérer la position Z de l'objet.

- void setX(double const x)
  - Permet de modifier la position X de l'objet.

- void setY(double const Y)
  - Permet de modifier la position Y de l'objet.

- void setZ(double const z)
  - Permet de modifier la position Z de l'objet.

Protected:

- double _x
  - Contient la valeur x de la position

- double _y
  - Contient la valeur y de la position

- double _z
  - Contient la valeur z de la position

Position.hpp (Prend une position X, Y, Z. Stock les position X, Y, Z

Texture.hpp

Methode :

- Texture (Color const & color = Color())
    - Constructeur par défaut.

- Texture (std::string const & sprite, Color const & color = Color(), int rotation = 0, Model3D const &model = Model3D())
    - Constructeur pour la texture de l'objet.

- ~Texture ()
    - Destructeur par défaut.

- Color const &getColor() const
    - Permet de récupérer la couleur de l'objet.

- void setColor(Color const &color)
    - Permet de modifier la couleur de l'objet.

- std::string const &getSprite() const
    - Permet de récupérer le sprite de l'objet.

- void setSprite(std::string const &sprite))
    - Permet modifier le sprite de l'objet.

- int const &getRotation() const
    - Permet de récupérer les positions de la rotation de l'objet.

- void setRotation(int rotation)
    - Permet de modifier les positions de la rotation de l'objet.

- Model3D const &getModel() const
    - Permet de récupérer le model 3D et ces informations.

- void setModel(Model3D const &model)
    - Permet de modifier le model 3D.

Private:

- Color _color
  - Contient les valeurs de couleurs RGBA.

- Std::string _sprite
  - Contient le nom des sprites.

- Model3D _model
  - Contient le model3D de l'objet.

- Int _rotation
  - Contient les positions de rotation de l'objet.

Texture.hpp (Permet de charger un sprite ou un model 3D ou une couleur ainsi que de choisir la rotation, stock sa couleur, son sprite, son model 3D et/ou sa rotation)

Arcade.hpp

Methode :

- Arcade ()
  - Constructeur par défaut.

- ~Arcade ()
  - Destructeur par défaut.

- void Setup(std::string const &startingLib)
  - Charge une Librairie.

- void Start()
  - Lance l'Arcade.

- void prevGraph() || void nextGraph()
  - Permet de charger la librairie précédente / suivante.

- void prevGame() | void nextGame()
  - Permet de charger le jeu précédent / suivant.

- void goUp() | void goDown() | void goLeft() | void goRight() | void pressEchap() | void shoot () || pressEight() | pressNine()
  - Permet d'utiliser les touches : Haut, Bas, Gauche, Droite, Echap, Espace, Huit, Neuf.

- void setInGameMode()
  - Permet de passer l'arcade en mode jeu.

- void setMenuMode()
  - Permet de passer l'arcade en mode menu.

- void initCurrentGame()
  - Initialise et lance un jeu.

- void setCurrentGameIndex(int gameIndex)
  - Modifie l'ID du jeu actuel pour en charger un autre.

- void setCurrentGame(game_ptr const &)
  - Charge un nouveau jeu.

- std::vector <game_ptr> getLibGames();
    - Récupère les librairies de tous les jeux et les insère dans un vector.

- std::vector<game_ptr> getlibGame() const
    - Récupère la librairie du jeu actuel.

- std::vector<std::string> getNameGame() const;
    - Récupère le nom de tous les jeux et les insère dans un vector.

Private:

- Arcade(const Arcade &obj)
    - Stock l'Arcade.

- void FillGameVector()
    - Remplis le vector de jeux avec les jeux présents.

- void FillGraphVector(std::string const &startingLib)
    - Remplis le vector de librairies avec les librairies présentes.

- bool _isRunning
    - Booléen permettant de connaitre l'état de l'arcade, True si l'arcade est en cours d'exécution, False si l'arcade ne s'exécute pas.

- int _frameRate
    - Défini le nombre de frame.

- IGraph* _currentGraph
    - Contient la librairie graphique en cours d'exécution.

- IGame* _currentGame
    - Contient le jeu en cours d'éxécution.

- std::vector<lib_ptr> _libGraph
    - Vector contenant toutes les librairies graphiques.

- std::vector<game_ptr> _libGame
    - Vector contenant toutes les libraires de jeu.

- std::vector<std::string> _nameGame
    - Vector contenant les noms de tous les jeux.

- std::vector<ScoreManager> _scoreVector
    - Vector contenant le score des joueurs.

- MenuController *_menu

    - Contient le menu.

- int _graphIndex()
    - Variable contenant l'index de la libraire en cours.

- int _gameIndex
    - Variable contenant l'index du jeu en cours.

- void transfertData() const
    - Fais le lien entre les IData du jeu et de la librairie graphique.

Arcade->IArcadeBridge (Arcade permet de Charger un jeu avec une library graphique donné, lancer le jeu, changer de library ou de jeu à tout moment, de gerer les fonctions de déplacement ainsi que les touches espace, entrée, echap, 8 et 9 du clavier, choisir l'état du jeu (Menu ou Jeu). Stock un IGraph de la library utilisé, un IGame du jeu utilisé, un vector des library graphique, un vector des jeux,un vector du nom des jeu, un vector des scores, le menu, un transfereData ainsi que le nombre de jeu et de library)

IArcadeBridge.hpp | INTERFACE

Methode :

- Virtual void prevGraph() || virtual void nextGraph()
    - Permet de charger la librairie précédente / suivante.

- Virtual void prevGame() | virtual void nextGame()
    - Permet de charger le jeu précédent / suivant.

- Virtual void goUp() | Virtual void goDown() | Virtual void goLeft() | Virtual void goRight() | Virtual void pressEchap() | Virtual void shoot () | Virtual pressEight() | Virtual pressNine()
    - Permet d'utiliser les touches : Haut, Bas, Gauche, Droite, Echap, Espace, Huit, Neuf.

IGame.hpp | INTERFACE

Methode :

- Virtual ~IGame ()
    - Destructeur par défaut.

- Virtual void initGame()
    - Initialise et lance un jeu choisis.

- virtual std::string const &getGameName() const = 0
    - Permet de récupérer le nom du jeu

- virtual int getFrameRatePerSecond() const = 0
    - Permet de récupérer le nombre de frame par seconde.

- virtual std::vector <AData *> const &getData() const = 0
    - Permer de récupérer et de stocker les AData dans un vector.

- virtual std::vector <std::string> const & getSprite() const = 0
    - Permet de récupérer les sprites et de les stocker dans un vector.

- void goUp() | void goDown() | void goLeft() | void goRight() | void pressEchap() | void shoot () || pressEight() | pressNine()
    - Permet d'utiliser les touches : Haut, Bas, Gauche, Droite, Echap, Espace, Huit, Neuf.

- virtual void play() = 0
    - Lance le jeu.

- virtual void getMap() = 0
    - Récupère la map.

virtual void whereIAm() = 0
    - Donne la position du joueur sur la map.

IGraph.hpp | INTERFACE

Methode :

- Virtual ~IGraph ()
    - Destructeur par défaut.

- Virtual void initLib()
    - Initialise et lance une librairie choisie.

- virtual std::string const &getLibName() const = 0
    - Permet de récupérer le nom de la librairie.

- virtual void giveData(std::vector <AData *> const &data) = 0
    - Permet de récupérer les AData dans un vector.

- virtual void giveSprite(std::vector <std::string> const &spriteList) = 0
    - Permet de récupérer les sprites stocker dans un vector.

- virtual void setBridge(IArcadeBridge * bridge) = 0
    - Permet de faire le lien entre la lib graphique et le jeu.

- virtual void handleData(AData const & data) = 0 | virtual void handleSphere(AData const & data) = 0 |virtual void handleCube(AData const & data) = 0 | virtual void handleCamera(AData const & data) = 0 |virtual void handleLight(AData const & data) = 0 |virtual void handleMusic(AData const & data) = 0 | virtual void handleText(AData const & data) = 0
    - Fonction de trie du AData.

- Virtual void prevGraph() = 0 || virtual void nextGraph() = 0
    - Permet de charger la librairie précédente / suivante.

- Virtual void prevGame() = 0| virtual void nextGame() = 0
    - Permet de charger le jeu précédent / suivant.

- Virtual void goUp() = 0 | virtual void goDown() = 0 |virtual void goLeft() = 0| virtual void goRight() = 0 |virtual void pressEchap() = 0 | virtual void shoot () = 0 || virtual pressEight() = 0 | virtual pressNine() = 0

    - Permet d’utiliser les touches : Haut, Bas, Gauche, Droite, Echap, Espace, Huit, Neuf.

- virtual void toggleRunning() const = 0
    - Permet d’activer ou de désactiver (ou de mettre en pause) le jeu.

LoadController.hpp | INTERFACE

Methode :

- ~LoadController ()
    - Destructeur par défaut.

- void LoadController()
    - Constructeur par défaut.

- static std::vector <std::string> readDirectory(std::string const &path, std::string const &startingLib)
    - Permet de lire le dossier et stock toutes les librairies dans un vector.