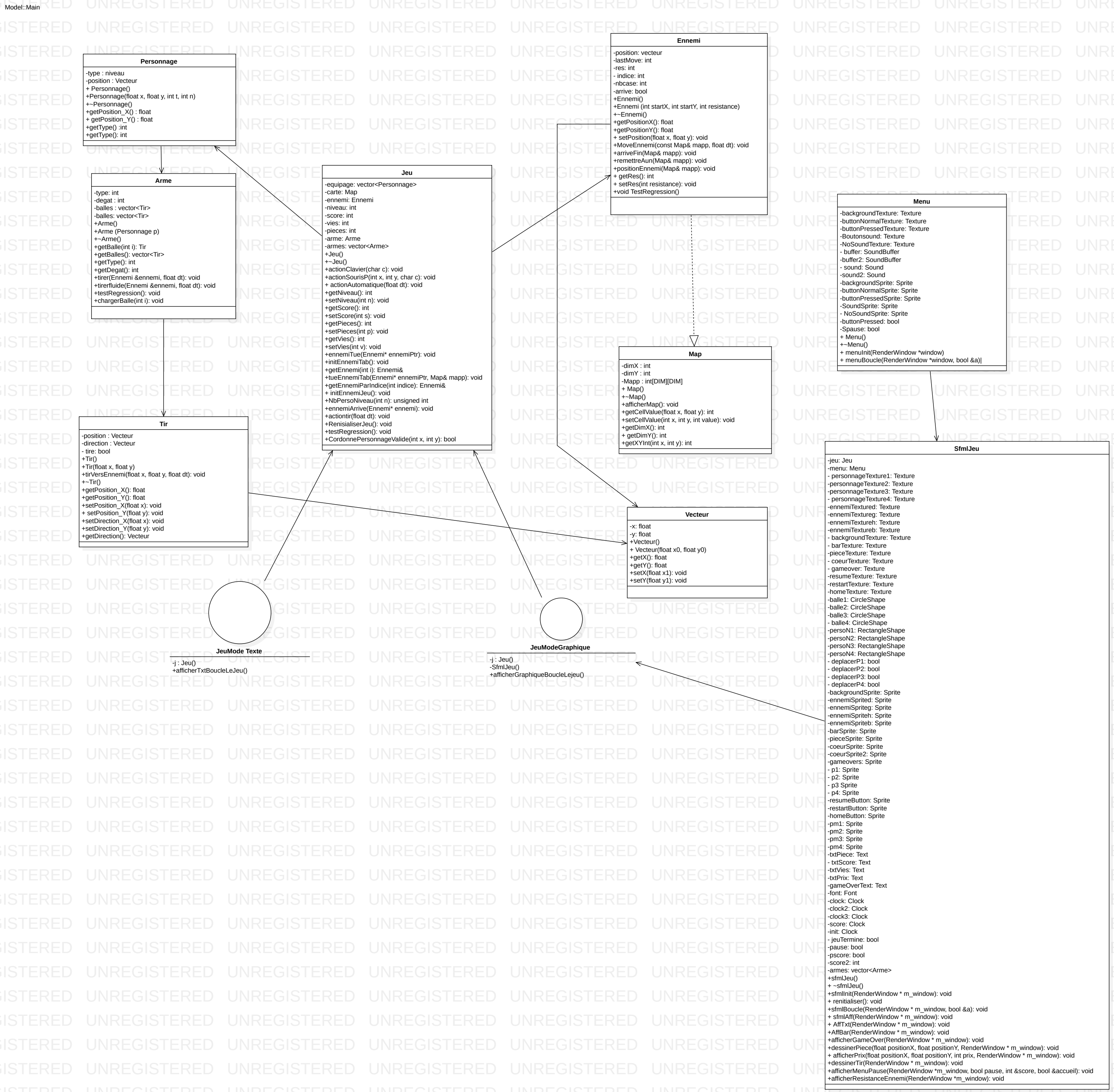

Model::Main

Personnage

-type : niveau
-position : Vecteur
+ Personnage()
+Personnage(float x, float y, int t, int n)
+~Personnage()
+getPosition_X() : float
+ getPosition_Y() : float
+getType() :int
+getType(): int

Arme

-type: int
-degat : int
-balles : vector<Tir>
-balles: vector<Tir>
+Arme()
+Arme (Personnage p)
+~Arme()
+getBalle(int i): Tir
+getBalles(): vector<Tir>
+getType(): int
+getDegat(): int
+tirer(Ennemi &ennemi, float dt): void
+tirerfluide(Ennemi &ennemi, float dt): void
+testRegression(): void
+chargerBalle(int i): void

Tir

-position : Vecteur
-direction : Vecteur
- tire: bool
+Tir()
+Tir(float x, float y)
+tirVersEnnemi(float x, float y, float dt): void
+~Tir()
+getPosition_X(): float
+getPosition_Y(): float
+setPosition_X(float x): void
+ setPosition_Y(float y): void
+setDirection_X(float x): void
+setDirection_Y(float y): void
+getDirection(): Vecteur

Jeu

-equipage: vector<Personnage>
-carte: Map
-ennemi: Ennemi
-niveau: int
-score: int
-vies: int
-pieces: int
-arme: Arme
-armes: vector<Arme>
+Jeu()
+~Jeu()
+actionClavier(char c): void
+actionSourisP(int x, int y, char c): void
+ actionAutomatique(float dt): void
+getNiveau(): int
+setNiveau(int n): void
+getScore(): int
+setScore(int s): void
+getPieces(): int
+setPieces(int p): void
+getVies(): int
+setVies(int v): void
+ennemiTue(Ennemi* ennemiPtr): void
+initEnnemiTab(): void
+getEnnemi(int i): Ennemi&
+tueEnnemiTab(Ennemi* ennemiPtr, Map& mapp): void
+getEnnemiParIndice(int indice): Ennemi&
+ initEnnemiJeu(): void
+NbPersoNiveau(int n): unsigned int
+ennemiArrive(Ennemi* ennemi): void
+actiontir(float dt): void
+RenisialiserJeu(): void
+testRegression(): void
+CordonnePersonnageValide(int x, int y): bool

Ennemi

-position: vecteur
-lastMove: int
-res: int
- indice: int
-nbcase: int
-arrive: bool
+Ennemi()
+Ennemi (int startX, int startY, int resistance)
+~Ennemi()
+getPositionX(): float
+getPositionY(): float
+ setPosition(float x, float y): void
+MoveEnnemi(const Map& mapp, float dt): void
+arriveFin(Map& mapp): void
+remettreAun(Map& mapp): void
+positionEnnemi(Map& mapp): void
+ getRes(): int
+ setRes(int resistance): void
+void TestRegression()

Map

-dimX : int
-dimY : int
-Mapp : int[DIM][DIM]
+ Map()
+~Map()
+afficherMap(): void
+getCellValue(float x, float y): int
+setCellValue(int x, int y, int value): void
+getDimX(): int
+ getDimY(): int
+getXYInt(int x, int y): int

Vecteur

-x: float
-y: float
+Vecteur()
+ Vecteur(float x0, float y0)
+getX(): float
+getY(): float
+setX(float x1): void
+setY(float y1): void

Menu

-backgroundTexture: Texture
-buttonNormalTexture: Texture
-buttonPressedTexture: Texture
-Boutonsound: Texture
-NoSoundTexture: Texture
- buffer: SoundBuffer
-buffer2: SoundBuffer
- sound: Sound
-sound2: Sound
-backgroundSprite: Sprite
-buttonNormalSprite: Sprite
-buttonPressedSprite: Sprite
-SoundSprite: Sprite
- NoSoundSprite: Sprite
-buttonPressed: bool
-Spause: bool
+ Menu()
+~Menu()
+ menuInit(RenderWindow *window)
+ menuBoucle(RenderWindow *window, bool &a)|

JeuMode Texte

-j : Jeu()
+afficherTxtBoucleLeJeu()

JeuModeGraphique

-j : Jeu()
-SfmlJeu()
+afficherGraphiqueBoucleLejeu()

SfmlJeu

-jeu: Jeu
-menu: Menu
- personnageTexture1: Texture
-personnageTexture2: Texture
-personnageTexture3: Texture
- personnageTexture4: Texture
-ennemiTextured: Texture
-ennemiTextureg: Texture
-ennemiTextureh: Texture
-ennemiTextureb: Texture
- backgroundTexture: Texture
- barTexture: Texture
-pieceTexture: Texture
- coeurTexture: Texture
- gameover: Texture
-resumeTexture: Texture
-restartTexture: Texture
-homeTexture: Texture
-balle1: CircleShape
-balle2: CircleShape
-balle3: CircleShape
- balle4: CircleShape
-persoN1: RectangleShape
-persoN2: RectangleShape
-persoN3: RectangleShape
-persoN4: RectangleShape
- deplacerP1: bool
- deplacerP2: bool
- deplacerP3: bool
- deplacerP4: bool
-backgroundSprite: Sprite
-ennemiSprited: Sprite
-ennemiSpriteg: Sprite
-ennemiSpriteh: Sprite
-ennemiSpriteb: Sprite
-barSprite: Sprite
-pieceSprite: Sprite
-coeurSprite: Sprite
-coeurSprite2: Sprite
-gameovers: Sprite
- p1: Sprite
- p2: Sprite
- p3 Sprite
- p4: Sprite
-resumeButton: Sprite
-restartButton: Sprite
-homeButton: Sprite
-pm1: Sprite
-pm2: Sprite
-pm3: Sprite
-pm4: Sprite
-txtPiece: Text
- txtScore: Text
-txtVies: Text
-txtPrix: Text
-gameOverText: Text
-font: Font
-clock: Clock
-clock2: Clock
-clock3: Clock
-score: Clock
-init: Clock
- jeuTermine: bool
-pause: bool
-pscore: bool
-score2: int
-armes: vector<Arme>
+sfmlJeu()
+ ~sfmlJeu()
+sfmlInit(RenderWindow * m_window): void
+ renitialiser(): void
+sfmlBoucle(RenderWindow * m_window, bool &a): void
+ sfmlAff(RenderWindow * m_window): void
+ AffTxt(RenderWindow * m_window): void
+AffBar(RenderWindow * m_window): void
+afficherGameOver(RenderWindow * m_window): void
+dessinerPiece(float positionX, float positionY, RenderWindow * m_window): void
+ afficherPrix(float positionX, float positionY, int prix, RenderWindow * m_window): void
+dessinerTir(RenderWindow * m_window): void
+afficherMenuPause(RenderWindow *m_window, bool pause, int &score, bool &accueil): void
+afficherResistanceEnnemi(RenderWindow *m_window): void