```
/**
 * \file Button.cc
 * \brief bouttons avec un texte
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 *
 */


#include <iostream>
#include <cstdlib>
#include <string>

#include "Button.h"
#include "image.h"

using namespace std;
using namespace sf;


/*##############################################################################
/#                                                                             #
/#            CONSTRUCTEUR & DESTRUCTEUR                                       #
/#                                                                             #
##############################################################################*/

Button::Button (int x, int y, int w, int h, bool press, string txt, int tx, int
ty, int tt )
: Element(x, y, w, h),
_press(false), _hover(false)
{
        _buttonImg = new Image();
        _font = new Font();

        loadimage( *_buttonImg, _buttonSprite, "images/button.png" );
        sizepositionimg(_buttonSprite, x, y, w, h);
        _font->LoadFromFile("images/antique.ttf");
        loadtextefont( _texte, _font, tt, x+tx, y+ty, txt);
}

Button::~Button ( )
{
        delete _font;
        delete _buttonImg;
}


/*##############################################################################
/#                                                                             #
/#            ACTION BUTTON                                                    #
/#                                                                             #
##############################################################################*/

/**
 * \fn Button::changeSprite(const float & x, const float & y, const float & w, c
onst float & h)
 * \brief change le carrÃ© de lecture du sprite sur l'image qu'il contient
 * \param x,y,w,h rÃ©els qui definissent les coordonnÃ©es et la taille du nouvea
u carrÃ© de lecture
 * \return rien
 */
void Button::changeSprite(const float & x, const float & y, const float & w, con
```



```
st float & h)
{
        setSprite( _buttonSprite, x, y, w, h);
        sizepositionimg(_buttonSprite, getX(), getY(), getW(), getH() );
}


/*##############################################################################
/#                                                                             #
/#            ACCESSEURS                                                       #
/#                                                                             #
##############################################################################*/

String Button::getTxt() const
{ return _texte; }

Sprite Button::getSprite() const
{ return _buttonSprite; }

bool Button::getPress() const
{ return _press; }

bool Button::getHover() const
{ return _hover; }

void Button::setTxt( const string & txt )
{ _texte.SetText(txt); }

void Button::setPress( const bool & p )
{ _press = p; }

void Button::setHover( const bool & h )
{ _hover = h; }
```

```
#ifndef BUTTON_H
#define BUTTON_H

/**
 * \file Button.h
 * \brief boutton avec un texte
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 *
 */

#include <SFML/Graphics.hpp>
#include <string>

#include "Element.h"

using namespace std;
using namespace sf;


/**
 * \class Button
 * \brief classe qui gÃ¨re les boutons
*/
class Button : public Element
{
        private:
                string _nomButton;
                Sprite _buttonSprite;
                String _texte;
                bool _press, _hover;
                Image* _buttonImg;
                Font* _font;

        public:

/// ######################## CONSTRUCTEUR DESTRUCTEUR ###################

                Button (int x, int y, int w, int h, bool press, string txt, int
tx, int ty, int tt );
                ~Button();

/// ######################## ACTION BUTTON ##############################

                void changeSprite(const float & x, const float & y, const float
& w, const float & h);

/// ######################## ACCESSEURS #################################

                String getTxt() const;
                Sprite getSprite() const;
                bool getPress() const;
                bool getHover() const;

                void setTxt( const string & txt );
                void setPress( const bool & p );
                void setHover( const bool & h );

};

#endif // BUTTON_H
```



```
/**
 * \file Cellule.cc
 * \brief Cellule du jeu
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 *
 */

#include "Cellule.h"
#include "util.h"

/*############################################################################
/#                                                                          #
/#            CONSTRUCTEUR & DESTRUCTEUR                                    #
/#                                                                          #
############################################################################*/

Cellule::Cellule( const int & x, const int & y, const float & aleac, const float
 & alear )
: Unite(x, y)
{
        _etatCellule = aleastate(aleac);
        _santeCellule = aleahealth(alear);
}

Cellule::Cellule( const int & x, const int & y, const state & e, const health &
s )
: Unite(x, y)
{
        _etatCellule = e;
        _santeCellule = s;
}

Cellule::~Cellule()
{}


/*############################################################################
/#                                                                          #
/#            ACCESSEURS                                                    #
/#                                                                          #
############################################################################*/

state Cellule::getEtat( ) const
{ return _etatCellule; }

health Cellule::getSante() const
{ return _santeCellule; }

void Cellule::setEtat (const state & e )
{ _etatCellule = e; }

void Cellule::setSante( const health & h )
{ _santeCellule = h; }
```

```
#ifndef CELLULE_H
#define CELLULE_H

/**
 * \file Cellule.h
 * \brief Cellule du jeu
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 *
 */

#include "Unite.h"
#include "enum.h"

/**
 * \class Cellule
 * \brief classe qui gÃ¨re une cellule
*/
class Cellule : public Unite
{
        private:
                state _etatCellule;
                health _santeCellule;

        public:

/// ######################## CONSTRUCTEUR DESTRUCTEUR ###################

                Cellule( const int & x, const int & y, const float & aleac, cons
t float & alear );
                Cellule( const int & x, const int & y, const state & e, const he
alth & s );
                ~Cellule();

/// ######################## ACCESSEURS #################################

                state getEtat() const;
                health getSante() const;
                void setEtat( const state & e );
                void setSante( const health & h );


};

#endif // CELLULE_H
```



```
/**
 * \file Element.cc
 * \brief element de classe
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 *
 */

#include "Element.h"

/*##############################################################################
/#                                                                             #
/#            CONSTRUCTEUR & DESTRUCTEUR                                       #
/#                                                                             #
##############################################################################*/

Element::Element(const float & x, const float & y, const float & w, const float
& h )
: _x(x), _y(y), _w(w), _h(h)
{}

Element::Element()
{}

Element::~Element ( )
{}


/*##############################################################################
/#                                                                             #
/#            ACCESSEURS LECTURE                                               #
/#                                                                             #
##############################################################################*/

float Element::getX ( ) const
{ return _x; }

float Element::getY ( ) const
{ return _y; }

float Element::getW ( ) const
{ return _w; }

float Element::getH( ) const
{ return _h; }


/*##############################################################################
/#                                                                             #
/#            ACCESSEURS ECRITURE                                              #
/#                                                                             #
##############################################################################*/

void Element::setX (const float & x )
{ _x = x; }

void Element::setY (const float & y )
{ _y = y; }

void Element::setW (const float & w )
{ _w = w; }
```

```
void Element::setH (const float & h )
{ _h = h; }
```



```
#ifndef ELEMENT_H
#define ELEMENT_H

/**
 * \file Element.h
 * \brief element de classe
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 *
 */

/**
 * \class Element
 * \brief classe mÃ¨re qui dÃ©finit un element
*/
class Element
{
        protected:

                float _x;
                float _y;
                float _w;
                float _h;

        public:

/// ######################## CONSTRUCTEUR DESTRUCTEUR ###################

                Element(const float & x, const float & y, const float & w, const
 float & h );
                Element();
                ~Element();

/// ######################## ACCESSEUR LECTURE ############################

                float getX() const;
                float getY() const;
                float getW() const;
                float getH() const;

/// ######################## ACCESSEUR ECRITURE ##########################

                void setX (const float & x );
                void setY (const float & y );
                void setW (const float & w );
                void setH (const float & h );
};

#endif // ELEMENT_H
```

```
#include "enum.h"

/**
 * \file enum.cc
 * \brief Ã©numÃ©rations
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 *
 */

state getState( const int & n )
{
        return state(n);
}

health getHealth( const int & n )
{
        return health(n);
}
```



```
#ifndef ENUM_H
#define ENUM_H

/**
 * \file enum.h
 * \brief Ã©numÃ©rations
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 *
 */

#include <string>

using namespace std;

/*****************************************************/
///         DEFINIT L'ETAT D'UNE CELLULE
/*****************************************************/

/**
 * \enum state
 * \brief Etat Cellule.
 *
 * state definit les trois etats possibles d'une cellule.
 * De plus chaque etat est associÃ© Ã  un entier.
 */
enum state { alive=1, dead=0, infest=-1 };

/*****************************************************/
///             DEFINIT LA RESISTANCE D'UNE CELLULE
/*****************************************************/

/**
 * \enum health
 * \brief sante Cellule.
 *
 * health definit si une cellule est resistante au attaque des virus
 * ou si il s'agit d'une cellule normale.
 * health est aussi associÃ© Ã  un entier.
 */
enum health { resist=1, normal=0 };


/*****************************************************/
///     DEFINIT LES ONGLETS DE LA FENETRE PARAMETRE
/*****************************************************/

/**
 * \enum tab
 * \brief Onglet du Menu.
 *
 * tab definit si l'onglet prÃ©sent dans MenuView concerne les cellules
 * ou concerne les virus;
 */
enum tab { cellule, virus };


/*****************************************************/
///         ACCESSEURS SUR LES ENUMS
/*****************************************************/

state getState( const int & n );
```

```
health getHealth( const int & n );

#endif
```



```
/**
 * \file event.cc
 * \brief gestion d'Ã©venements
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 *
 */

#include <iostream>

#include "event.h"

using namespace std;


/*##############################################################################
/#                                                                             #
/#            EVENEMENTS DE BASE                                               #
/#                                                                             #
/# ces fonctions simplifie les evenement basique proposÃ© par SFML              #
                                                #
/#                                                                             #
##############################################################################*/

/// ########## Evenement sourie ############

/**
 * \fn clickDownLeft( Event* event )
 * \param event pointeur d'Event provenant de MenuView et contenant les Ã©venmen
ts sourie
 * \return un boolÃ©en qui indique si on effectue un click prÃ©ssÃ© gauche
 */
bool clickDownLeft( Event* event )
{
        return ( (event->Type == sf::Event::MouseButtonPressed)
                        && (event->MouseButton.Button == sf::Mouse::Left) );
}

/**
 * \fn clickDownRight( Event* event )
 * \param event pointeur d'Event provenant de MenuView et contenant les Ã©venmen
ts sourie
 * \return un boolÃ©en qui indique si on effectue un click prÃ©ssÃ© droit
 */
bool clickDownRight( Event* event )
{
        return ( (event->Type == sf::Event::MouseButtonPressed)
                        && (event->MouseButton.Button == sf::Mouse::Right) );
}

/**
 * \fn clickUpLeft( Event* event )
 * \param event pointeur d'Event provenant de MenuView et contenant les Ã©venmen
ts sourie
 * \return un boolÃ©en qui indique si on effectue un click relachÃ© gauche
 */
bool clickUpLeft( Event* event )
{
        return ( (event->Type == sf::Event::MouseButtonReleased)
                        && (event->MouseButton.Button == sf::Mouse::Left) );
}
```

```
/**
 * \fn clickUpRight( Event* event )
 * \param event pointeur d'Event provenant de MenuView et contenant les Ã©venmen
ts sourie
 * \return un boolÃ©en qui indique si on effectue un click relachÃ© droit
 */
bool clickUpRight( Event* event )
{
        return ( (event->Type == sf::Event::MouseButtonReleased)
                        && (event->MouseButton.Button == sf::Mouse::Right) );
}

/// ############## Evenement clavier ##################

/**
 * \fn keyPress( Event* event, const Key::Code & key )
 * \param event pointeur d'Event provenant de MenuView et contenant les Ã©venmen
ts clavier
 * \param key Key::Code prÃ©cise la touche qui nous interesse
 * \return un boolÃ©en qui indique si la touche en question est prÃ©ssÃ©
 */
bool keyPress( Event* event, const Key::Code & key )
{
        return ( (event->Type == sf::Event::KeyPressed) && (event->Key.Code == k
ey) );
}

/*#########################################################################################
#                                                                                        #
#            EVENEMENTS SUR DES OBJETS                                                   #
#                                                                                        #
#########################################################################################*/

/**
 * \fn clickButton(Event* event, const float & mx, const float & my, Button* but
ton)
 * \brief Fonction qui gÃ©re les clicks et selection de l'objet Button
 * \param event pointeur d'Event provenant de MenuView et contenant les Ã©venmen
ts sourie
 * \param button pointeur sur Boutton, pour faire appel au diffÃ©rent accesseur
de la classe
 * \param mx et my rÃ©els qui indique les coordonnÃ©es de la sourie
 * \return un boolÃ©en qui indique si ont peut executer l'action que produit le
boutton
 */
bool clickButton(Event* event, const float & mx, const float & my, Button* butto
n)
{
// si on click gauche sur un boutton
        if ( clickDownLeft(event) && !button->getPress()
                && mx >= button->getX() && mx <= ( button->getX() + button->getW
())
                && my >= button->getY() && my <= ( button->getY() + button->getH
()) )
        {
                button->setPress(true);
        }
// si on relache la sourie sur la zone du boutton
        if ( clickUpLeft(event) && button->getPress()
                && mx >= button->getX() && mx <= ( button->getX() + button->getW
())
                && my >= button->getY() && my <= ( button->getY() + button->getH
```



```
()) )
        {
                button->setPress(false);
                return true;
        }
// si on relache la sourie hors du boutton
        else if ( clickUpLeft(event) && button->getPress() )
        {
                button->setPress(false);
        }

        else if ( !clickUpLeft(event) && !clickDownLeft(event)
                && mx >= button->getX() && mx <= ( button->getX() + button->getW
())
                && my >= button->getY() && my <= ( button->getY() + button->getH
()) )
        {
                button->setHover(true);
                return false;
        }
        else
        {
                button->setHover(false);
                return false;
        }
}

/**
 * \fn moveSlider(Event* event, Element* element, Slider* slider, const float &
mx, const float & my)
 * \brief Fonction de gestion du mouvement d'un objet Slider
 * \param event pointeur d'Event provenant de MenuView et contenant les Ã©venmen
ts sourie
 * \param element pointeur sur l'Element bougeable du Slider pour rÃ©cupÃ©rer la
 taille et coordonnÃ©es
 * \param slider pointeur sur l'objet Slider, permet de faire appel au mÃ©thode
contenue dans le slider
 * \param mx et my rÃ©els qui indique les coordonnÃ©es de la sourie
 * \return rien.
 */
void moveSlider(Event* event, Element* element, Slider* slider, const float & mx
, const float & my)
{
        if ( clickDownLeft(event)
                && mx >= element->getX() && mx <= ( element->getX() + element->g
etW()+2)
                && my >= element->getY() && my <= ( element->getY() + element->g
etH()) )
        {
                slider->setMove( true );        // le slider est en Ã©tat de mou
vement

        }
// quand on relache le click et que le slider est en etat de bougÃ©
        else if ( clickUpLeft(event) && slider->getMove() )
        {
                slider->setMove( false );
        }
// si on click sur la bar du slider, hors de la partie mobile
        if ( clickDownLeft(event)
                && !(mx >= element->getX() && mx <= ( element->getX() + element-
>getW()+2)
```

```
                && my >= element->getY() && my <= ( element->getY() + element->g
etH()))
                && (mx >= slider->getX() && mx <= ( slider->getX() + slider->get
W())
                && my >= slider->getY() && my <= ( slider->getY() + slider->getH
()))  )
        {
                element->setX( mx );            // definit la position du slider
 par rapport Ã  sourie
                slider->updateSlider();         // met Ã  jour le slider pour al
igner et vÃ©rifier la position
        }

        if ( slider->getMove() )
        {
                element->setX( mx );            // definit la position du slider
 par rapport Ã  sourie
                slider->updateSlider();         // met Ã  jour le slider pour al
igner et vÃ©rifier la position
        }
}
```



```
#ifndef EVENT_H
#define EVENT_H

/**
 * \file event.h
 * \brief gestion d'Ã©venements
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 *
 */

#include <SFML/Graphics.hpp>

#include "Button.h"
#include "Element.h"
#include "Slider.h"

using namespace sf;

/// ###################### EVENEMENT DE BASE ##################

        // evenement sourie

bool clickDownLeft( Event* event );
bool clickDownRight( Event* event );
bool clickUpLeft( Event* event );
bool clickUpRight( Event* event );

        // evenement clavier

bool keyPress( Event* event, const Key::Code & key );

/// ###################### EVENEMENT SUR OBJETS ##################

bool clickButton(Event* event, const float & mx, const float & my, Button* butto
n);
void moveSlider(Event* event, Element* element, Slider* slider, const float & mx
, const float & my);


#endif // EVENT_H
```

```
/**
 * \file GameModel.cc
 * \brief model du jeu
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 *
 */

#include <iostream>

#include "GameModel.h"
#include "enum.h"
#include "event.h"

using namespace std;
//~ using namespace sf;

/*#########################################################################################
/#                                                                                       #
/#            CONSTRUCTEUR & DESTRUCTEUR                                                 #
/#                                                                                       #
#########################################################################################*/

/**
 * \fn GameModel::GameModel (int w, int h )
 * \brief constructeur par paramÃ©tre du GameModel
 * \param w,h des rÃ©els pour indiquÃ© la taille de la fenÃªtre
 */
GameModel::GameModel(const float & w, const float & h )
: _w(w), _h(h), _stateGame( false), _clickL(false), _clickR(false)
{
        _gVirus = new GroupeVirus( 2, 6.0, 1);
        _gCellule = new GrilleCellule(2,3,3,3);
        _gCellule->update(_w,_h);
        _vitesseExe = 0;
        _timeVitesse.Reset();
        _save = new GameSave();
}

GameModel::~GameModel ( )
{
        delete _save;
        delete _gVirus;
        delete _gCellule;
}

/*#########################################################################################
/#                                                                                       #
/#            ACTION SUR LE JEU                                                          #
/#                                                                                       #
#########################################################################################*/

/**
 * \fn GameModel::nextStep()
 * \brief applique les changements Ã  l'Ã©tape suivante
 * \param rien
 * \return rien
 */
void GameModel::nextStep()
{
        _gVirus->lifeTimeVirus(_stateGame);
```



```
        if ( _timeVitesse.GetElapsedTime() >= _vitesseExe )
        {
                _timeVitesse.Reset();
                if ( _stateGame )
                {
                        _gVirus->moveVirus();
                        _gCellule->loi_de_vie();
                        virusVsCellule();
                }
        }
}

/**
 * \fn GameModel::play_pause(const bool & p)
 * \brief mÃ©thode qui met le jeu en pause ou en lecture
 * \param p boolÃ©en qui prÃ©cise si le bouton play pause Ã  Ã©tÃ© prÃ©ssÃ©
 * \return rien
 */
void GameModel::play_pause(const bool & p)
{
        if ( p )
        {
                if ( _stateGame ) _stateGame = false;
                else _stateGame = true;
        }
}

/**
 * \fn GameModel::virusVsCellule()
 * \brief fait la confrontation des Virus sur les Cellules
 * \param rien
 * \return rien
 */
void GameModel::virusVsCellule()
{
        int ti = _gVirus->getSizeGroupe();
        int tci, tcj, ci, cj;
        _gCellule->getTailleGrille(tci, tcj);
        float vx, vy, vw, vh;

        for (int i=0; i<ti; i++)
        {
                _gVirus->getCoordonne(i, vx, vy, vw, vh);
                ci = ( vx * tci ) / _w;
                cj = ((vy * tcj ) / _h)-1;

                if (  !_gVirus->getTargetVirus(i) )
                {

// si la position du virus est bien situÃ© dans la fenÃªtre de jeu

                        if ( vx >= 0 && vx <= (_w - vw) && vy > 30 && vy < ((_h+
30) - vh)
                                && ci >= 0 && ci < tci && cj >= 0 && cj < tcj )
                        {
                                if ( _gCellule->getEtatCellule(ci, cj) == alive
&& _gCellule->getHealthCellule(ci, cj) == normal
                                        && _stateGame)
                                {
                                        _gCellule->setEtatCellule(ci, cj, infest
);
                                        _gVirus->setCoordonneVirus(i, ci*(_w/tci
```

```
)+((_w/tci)/2)-(vw/2), cj*(_h/tcj)+((_h/tcj)/2)-(vh/2) );
                                        _gVirus->setTargetVirus(i, true);
                                        _gVirus->setResetIncubVirus( i );
                                }
                        }
// si le virus sort de la fenetre
                        else
                        {
                                _gVirus->delVirus( i );
                        }
                }

// si le virus a deja une cible, il applique son temps d'incubation
                else
                {
                        //~ if ( ci >= 0 && ci < tci && cj >= 0 && cj < tcj )
                                //~ _gCellule->setEtatCellule(ci, cj, infest);
                        _gVirus->incubTimeVirus( _stateGame, i );

                }
        }
}

/**
 * \fn GameModel::loadSave( const bool & b )
 * \brief charge la derniere sauvegarde
 * \param b boolÃ©en qui indique si le bouton a Ã©tÃ© prÃ©ssÃ©
 * \return rien
 */
void GameModel::loadSave( const bool & b )
{ if ( b ) _save->open("save/game.sav", _gCellule, _gVirus, _stateGame);  }

/**
 * \fn GameModel::saveSave( const bool & b )
 * \brief sauvegarde la partie en cours
 * \param b boolÃ©en qui indique si le bouton a Ã©tÃ© prÃ©ssÃ©
 * \return rien
 */
void GameModel::saveSave( const bool & b )
{ if ( b ) _save->save("save/game.sav", _gCellule, _gVirus, _stateGame); }


/*#############################################################################
/#                                                                           #
/#            ACTION SUR CELLULES                                            #
/#                                                                           #
#############################################################################*/

/**
 * \fn GameModel::add_column (const bool & add )
 * \brief ajoute une colonne de Cellule
 * \param add boolÃ©en qui indique si le bouton a Ã©tÃ© prÃ©ssÃ©
 * \return rien
 */
void GameModel::add_column (const bool & add )
{
        if ( add )
        {
                int ti, tj;
                _gCellule->add_column(_w, _h);

                getTailleGrilleCellule(ti, tj);
```



```
                _gVirus->updateVirus((_w/ti)/3, (_h/tj)/3, -ti, 0 );
        }

}

/**
 * \fn GameModel::add_line(const bool & add)
 * \brief ajoute une ligne de Cellule
 * \param add boolÃ©en qui indique si le bouton a Ã©tÃ© prÃ©ssÃ©
 * \return rien
 */
void GameModel::add_line(const bool & add)
{
        if ( add )
        {
                _gCellule->add_line(_w, _h);

                int ti, tj;
                getTailleGrilleCellule(ti, tj);
                _gVirus->updateVirus((_w/ti)/3, (_h/tj)/3, 0, -tj );
        }
}

/**
 * \fn GameModel::remove_column(const bool & del)
 * \brief enleve une colonne de Cellule
 * \param del boolÃ©en qui indique si le bouton a Ã©tÃ© prÃ©ssÃ©
 * \return rien
 */
void GameModel::remove_column(const bool & del)
{
        if ( del )
        {
                bool up;
                int ti, tj;

                up = _gCellule->remove_column(_w, _h);
                getTailleGrilleCellule(ti, tj);
                 if ( up )
                        _gVirus->updateVirus((_w/ti)/3, (_h/tj)/3, ti, 0 );
        }
}

/**
 * \fn GameModel::remove_line(const bool & del)
 * \brief enleve une ligne de Cellule
 * \param del boolÃ©en qui indique si le bouton a Ã©tÃ© prÃ©ssÃ©
 * \return rien
 */
void GameModel::remove_line(const bool & del)
{
        if ( del )
        {
                int ti, tj;
                bool up;

                up = _gCellule->remove_line(_w, _h);
                getTailleGrilleCellule(ti, tj);
                if ( up )
                        _gVirus->updateVirus((_w/ti)/3, (_h/tj)/3, 0, tj );
        }
}
```

```
/**
 * \fn GameModel::alive_cellule( Event* event, const int & mx, const int & my )
 * \brief rend une cellule vivante
 * \param event Event contenant l'evenement sourie du jeu
 * \param mx, my entier qui dÃ©finissent les coordonnÃ©e de la sourie
 * \return rien
 */
void GameModel::alive_cellule( Event* event, const int & mx, const int & my )
{
        if ( clickDownLeft(event))
                _clickL = true;
        if ( clickUpLeft(event) )
                _clickL = false;
        if ( _clickL && mx > 0 && mx < _w && my > 30 && my < _h+30)
        {
                int ti, tj, i, j;
                _gCellule->getTailleGrille(ti, tj);
                i = ( mx * ti ) / _w;
                j = ((my-30) * tj) / _h ;
                _gCellule->setEtatCellule(i, j, alive);
        }
}

/**
 * \fn GameModel::dead_cellule( Event* event, const int & mx, const int & my )
 * \brief rend une cellule morte
 * \param event Event contenant l'evenement sourie du jeu
 * \param mx, my entier qui dÃ©finissent les coordonnÃ©e de la sourie
 * \return rien
 */
void GameModel::dead_cellule( Event* event, const int & mx, const int & my )
{
        if ( clickDownRight(event) )
                _clickR = true;
        if ( clickUpRight(event) )
                _clickR = false;
        if ( _clickR && mx > 1 && mx < _w && my > 30 && my < _h+30 )
        {
                int ti, tj, i, j;
                _gCellule->getTailleGrille(ti, tj);
                i = ( mx * ti ) / _w;
                j = ((my-30) * tj) / _h ;
                _gCellule->setEtatCellule(i, j, dead);
        }
}

/**
 * \fn GameModel::update_all_Cellule(const bool & b)
 * \brief fait appel Ã  la mise a jours des Cellules
 * \param b boolÃ©en qui indique si le bouton a Ã©tÃ© prÃ©ssÃ©
 * \return rien
 */
void GameModel::update_all_Cellule(const bool & b)
{
        if ( b )
                _gCellule->updateAll();
}


/*#########################################################################
/#                                                                       #
```



```
/#            ACTION SUR VIRUS                                           #
/#                                                                       #
#########################################################################*/


/**
 * \fn GameModel::add_virus ( const bool & add )
 * \brief ajoute un virus dans le jeu
 * \param add boolÃ©en qui indique si le bouton a Ã©tÃ© prÃ©ssÃ©
 * \return rien
 */
void GameModel::add_virus ( const bool & add )
{ if ( add ) _gVirus->addVirus(); }

/**
 * \fn GameModel::del_virus ( const bool & del )
 * \brief enleve le dernier virus ajouter dans le jeu
 * \param del boolÃ©en qui indique si le bouton a Ã©tÃ© prÃ©ssÃ©
 * \return rien
 */
void GameModel::del_virus ( const bool & del )
{ if ( del ) _gVirus->removeVirus(); }

/**
 * \fn GameModel::removeAllVirus ( const bool & all )
 * \brief enlÃ©ve tous les Virus du Jeu
 * \param all boolÃ©en qui indique si le bouton a Ã©tÃ© prÃ©ssÃ©
 * \return rien
 */
void GameModel::removeAllVirus ( const bool & all )
{ if ( all ) _gVirus->removeAll(); }

/**
 * \fn GameModel::removeVirus( const int & i, const bool & d )
 * \brief enleve un Virus prÃ©cis
 * \param d boolÃ©en qui indique si le bouton a Ã©tÃ© prÃ©ssÃ©
 * \param i entier qui indique l'indice du virus a supprimÃ©
 * \return rien
 */
void GameModel::removeVirus( const int & i, const bool & d )
{ if ( d ) _gVirus->delVirus(i); }


/*#########################################################################
/#                                                                       #
/#            ACCESSEURS SUR VIRUS                                       #
/#                                                                       #
#########################################################################*/

int GameModel::getSizeGroupeVirus() const
{ return _gVirus->getSizeGroupe(); }

void GameModel::getCoordonneVirus( const int & i, float & x, float & y, float &
w, float & h ) const
{ _gVirus->getCoordonne(i, x, y, w, h); }

void GameModel::setNbEfantVirus( const int & e )
{ _gVirus->setNbEfant(e); }

void GameModel::setDureVieVirus( const float & d )
{ _gVirus->setDureVie(d); }
```

```
void GameModel::setDureIncubVirus( const float & d )
{ _gVirus->setDureIncub(d); }


/*############################################################################
/#                                                                          #
/#            ACCESSEURS SUR CELLULE                                        #
/#                                                                          #
############################################################################*/

void GameModel::getCoordonneCellule( const int & i, const int & j, Unite & e) co
nst
{ _gCellule->getCoordonne(i, j, e); }

void GameModel::getSizePictureCellule( float & w, float & h ) const
{ _gCellule->getSizePicture(w,h); }

health GameModel::getSanteCellule(const int & i, const int & j) const
{ _gCellule->getHealthCellule(i, j); }

state GameModel::getEtatCellule(const int & i, const int & j) const
{ return _gCellule->getEtatCellule(i,j); }

void GameModel::getTailleGrilleCellule( int & ti, int & tj ) const
{ _gCellule->getTailleGrille(ti, tj); }

void GameModel::setMinAliveCellule( const int & a )
{ _gCellule->setMinAlive(a); }

void GameModel::setMaxAliveCellule( const int & a )
{ _gCellule->setMaxAlive(a); }

void GameModel::setMinDeadCellule( const int & a )
{ _gCellule->setMinDead(a); }

void GameModel::setMaxDeadCellule( const int & a )
{ _gCellule->setMaxDead(a); }

void GameModel::setAleaCelluleGrille( const float & a )
{ _gCellule->setAleaCellule(a); }

void GameModel::setAleaResistGrille( const float & a )
{ _gCellule->setAleaResist(a); }


/*############################################################################
/#                                                                          #
/#            ACCESSEURS SUR MODEL                                          #
/#                                                                          #
############################################################################*/

bool GameModel::getStateGame() const
{ return _stateGame; }

int GameModel::getVitesse() const
{ return (100 - _vitesseExe)*100; }

void GameModel::setW(const int & w )
{ _w = w; }


void GameModel::setH(const int & h )
```



```
{ _h = h; }

void GameModel::setVitesse( const int & v)
{ _vitesseExe = ( 100 - (v*1.0) ) / 100; }
```

```
#ifndef GAMEMODEL_H
#define GAMEMODEL_H



#include <cstdlib>
#include <string>
#include <vector>

#include "GameSave.h"
#include "Unite.h"
#include "GrilleCellule.h"
#include "GroupeVirus.h"
#include "enum.h"

using namespace std;

/**
 * \class GameModel
 * \brief classe qui gÃ¨re la partie model
*/
class GameModel
{
        private:
                float _w;
                float _h;
                bool _stateGame, _clickL, _clickR;
                GrilleCellule * _gCellule;
                GroupeVirus * _gVirus;
                float _vitesseExe;
                GameSave * _save;
                Clock _timeVitesse;

        public:

/// ###################### CONSTRUCTEUR & DESTRUCTEUR ##################

                GameModel(const float & w, const float & h );
                ~GameModel();

/// ###################### ACTION SUR LE JEU ############################

                void nextStep();
                void play_pause(const bool & p);
                void virusVsCellule();
                void loadSave( const bool & b );
                void saveSave( const bool & b );

/// ###################### ACTION SUR CELLULES ##########################

                void add_column( const bool & add);
                void add_line(const bool & add);
                void remove_column(const bool & del);
                void remove_line(const bool & del);
                void alive_cellule( Event* event, const int & mx, const int & my
 );
                void dead_cellule( Event* event, const int & mx, const int & my
);
                void update_all_Cellule(const bool & b);

/// ###################### ACTION SUR VIRUS #############################
```



```
                void add_virus ( const bool & add );
                void del_virus ( const bool & del );
                void removeAllVirus ( const bool & all );
                void updateVirus(const int & w, const int & h);
                void removeVirus( const int & i, const bool & d );


/// ###################### ACCESSEUR DU GROUPE VIRUS ####################

                int getSizeGroupeVirus() const;
                void getCoordonneVirus( const int & i, float & x, float & y, flo
at & w, float & h ) const;
                void setNbEfantVirus( const int & e );
                void setDureVieVirus( const float & d );
                void setDureIncubVirus( const float & d );

/// ###################### ACCESSEUR DU GRILLE CELLULES #################


                state getEtatCellule(const int & i, const int & j) const;
                health getSanteCellule(const int & i, const int & j) const;
                void getCoordonneCellule( const int & i, const int & j, Unite &
e) const;
                void getSizePictureCellule( float & w, float & h ) const;
                void getTailleGrilleCellule( int & ti, int & tj ) const;
                void setMinAliveCellule( const int & a );
                void setMaxAliveCellule( const int & a );
                void setMinDeadCellule( const int & a );
                void setMaxDeadCellule( const int & a );
                void setAleaCelluleGrille( const float & a );
                void setAleaResistGrille( const float & a );

/// ###################### ACCESSEUR DU MODEL ###########################

                bool getStateGame() const;
                int getVitesse() const;
                void setW(const int & w );
                void setH(const int & h );
                void setVitesse( const int & v);

};

#endif // GAMEMODEL_H
```

mai 19, 11 23:22 **GameSave.cc** Page 1/1

```
/**
 * \file GameSave.cc
 * \brief gestion de sauvegarde
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 */


#include <fstream>
#include <iostream>

#include "GameSave.h"

using namespace std;

GameSave::GameSave()
{ }

 GameSave::~GameSave()
{ }

/**
 * \fn GameSave::open(string nom_sauvegarde, GrilleCellule* grille, GroupeVirus*
 groupe, bool & s)
 * \brief ouvre une sauvegarde
 * \param grille GrilleCellule Ã  modifier
 * \param groupe GroupeVirus Ã  modifier
 * \param s boolÃ©en definissant l'etat du jeu a modifier
 * \return rien
 */
void GameSave::open(string nom_sauvegarde, GrilleCellule* grille, GroupeVirus* g
roupe, bool & s )
{
        fstream f;
        f.open( nom_sauvegarde.data(), ios::in );

        grille->lireGrilleCellule(f);
        groupe->lireGroupeVirus(f);
        f >> s;
}

/**
 * \fn GameSave::save(string nom_sauvegarde, GrilleCellule* grille, GroupeVirus*
 groupe, const bool & s)
 * \brief sauvegarde le jeu en cours
 * \param grille GrilleCellule Ã  enregistrer
 * \param groupe GroupeVirus Ã  enregistrer
 * \param s boolÃ©en definissant l'etat du jeu Ã  enregistrer
 * \return rien
 */
void GameSave::save(string nom_sauvegarde, GrilleCellule* grille, GroupeVirus* g
roupe, const bool & s)
{
        fstream f;
        f.open( nom_sauvegarde.data(), ios::out );

        grille->ecrireGrilleCellule(f);
        groupe->ecrireGroupeVirus(f);
        f << s << endl;

        f.close();
}
```

mai 19, 11 23:22 **GameSave.h** Page 1/1

```
#ifndef GAMESAVE_H
#define GAMESAVE_H

/**
 * \file GameSave.h
 * \brief gestion de sauvegarde
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 */

#include <string>

#include "GrilleCellule.h"
#include "GroupeVirus.h"

using namespace std;

/**
 * \class GameSave
 * \brief classe qui sauvegarde une partie
*/
class GameSave
{
        private:
                int _numSave;

        public:
                GameSave ( );
                ~GameSave ( );

                void open(string nom_sauvegarde, GrilleCellule* grille, GroupeVi
rus* groupe, bool & s );
                void save(string nom_sauvegarde, GrilleCellule* grille, GroupeVi
rus* groupe, const bool & s );
};

#endif // GAMESAVE_H
```

```
/**
 * \file GameView.cc
 * \brief View du jeu
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 */

#include <iostream>
#include <SFML/Graphics.hpp>
#include <SFML/Audio.hpp>

#include "GameView.h"
#include "Button.h"
#include "image.h"
#include "util.h"
#include "event.h"

using namespace std;
using namespace sf;

static sf::Font* _font;


/*#########################################################################################
/#                                                                                       #
/#            CONSTRUCTEUR & DESTRUCTEUR                                                 #
/#                                                                                       #
#########################################################################################*/

/**
 * \fn GameView::GameView(const float & w, const float & h, const float & x, con
st float & y )
 * \brief constructeur par paramÃ©tre du GameView
 * \param w,h des rÃ©els pour indiquÃ© la taille de la fenÃªtre
 * \param x,y des rÃ©els pour definir la position de la fenÃªtre
 */
GameView::GameView(const float & w, const float & h, const float & x, const floa
t & y )
: Element(x, y, w, h),
_buttonPar( 680, 0, 120, 30, 0, "Parametre", 12, 1, 20),
_open(380, 0, 100, 30, 0, "Ouvrir", 22, 1, 20),
_save(490, 0, 100, 30, 0, "Charger", 18, 1, 20)

{
        _window = new RenderWindow(sf::VideoMode(w, h, 32), "Game Of Life", sf::St
yle::Close );  //| sf::Style::Resize
        _window->SetPosition(x,y);

        loadimage(_bgImg, _bgSprite, "images/bg.png",0,0,800,630);
        loadimage(_bgBar, _barSprite,"images/bg_bar.png");
        loadimage(_celluleImg, _celluleSprite, "images/cell.png");
        loadimage(_virusImg, _virusSprite, "images/virus.png");
        loadimage(_selectImg, _selectSprite, "images/rect.png");

        _font = new Font();
        _font->LoadFromFile("images/antique.ttf");

        loadtextefont( _nbCellule, _font, 17, 5, 4, "Cellule : ");
        loadtextefont( _nbVirus, _font, 17, 200, 4, "Virus : ");

        loadtextefont( _valCellule, _font, 17, 70, 4, "0" );
        loadtextefont( _valVirus, _font, 17, 260, 4, "0" );
```



```
}

 GameView::~GameView ( )
{
        if(_window!= NULL)
        {
        delete _window;
        }
    delete _font;
}


/*#########################################################################################
/#                                                                                       #
/#            ACTION FENETRE DE JEU                                                      #
/#                                                                                       #
#########################################################################################*/

/**
 * \fn GameView::updateVal()
 * \brief met a jour le texte d'information des Cellules et Virus
 * \param rien
 * \return rien
 */
void GameView::updateVal()
{
        int vi, ci, cj;
        _model->getTailleGrilleCellule(ci, cj);
        vi = _model->getSizeGroupeVirus();

        _valVirus.SetText( toString(vi)  );
        _valCellule.SetText( toString(cj) + " X " + toString(ci) );
}

/**
 * \fn GameView::draw()
 * \brief dessine les objets sur la fenetre de jeu
 * \param rien
 * \return rien
 */
void GameView::draw()
{
        //~ int ti, tj; _model->getTailleGrilleCellule(ti,tj);

        _window->Draw(_bgSprite);

        afficherCellules(_model, _window, &_celluleSprite);
        afficherVirus(_model, _window, &_virusSprite );

// ~~~~~~~~~~~~ AFFICHAGE DU TEXTE ~~~~~~~~~~~~~~~~~~~~~~

        _window->Draw(_barSprite);
        afficherButton( _window, &_buttonPar );
        afficherButton( _window, &_open );
        afficherButton( _window, &_save );

        _window->Draw( _nbCellule );
        _window->Draw( _nbVirus );

        if ( _mouseX >= 0 && _mouseX < _w && _mouseY >= 30 && _mouseY < _h )
                //~ afficherselect( _window, &_selectSprite,  ( _mouseX * ti ) /
```

```
 _w, (_mouseY * tj )/ (_h-30.0), _w/ti, (_h-30)/tj);

        updateVal();
        _window->Draw( _valCellule );
        _window->Draw( _valVirus );

        _window->Display();

// permet d'avoir un usleep constant en fonction de si la fenetre est ouvert ou
pas
        if ( _menu->getWindow() )
                usleep(10000);
        else
                usleep(20000);
}

/**
 * \fn GameView::treatEvents()
 * \brief traite les Ã©venements qu'effectue le joueur
 * \param rien
 * \return rien
 */
bool GameView::treatEvents()
{
        bool result = false;
        if(_window->IsOpened())
        {
        result = true;
        sf::Event event;

                while ( _window->GetEvent(event) )
                {

// ~~~~~~~~~ FERMER LA FENETRE DE JEU ~~~~~~~~~~~~~~~~~~~~~
                        Key::Code key = Key::Escape;


                if ((event.Type == Event::Closed) || keyPress( &event, Key::Esca
pe) )
                        {
                                result = false;
                }

// ~~~~~~~~~ RECUPERE COORDONNE SOURIE ~~~~~~~~~~~~~~~~~~~~

                _mouseX = _window->GetInput().GetMouseX();
                _mouseY = _window->GetInput().GetMouseY();


// ~~~~~~~~~ DONNER VIE OU MORT D'UNE CELLULE ~~~~~~~~~~~~~~

                _model->alive_cellule( &event, _mouseX, _mouseY);
                _model->dead_cellule( &event, _mouseX, _mouseY);


// ~~~~~~~~~~ OUVRIR FENETRE DE PARAMETRE ~~~~~~~~~~~~~~~~~

                        _menu->openWindow( clickButton( &event, _mouseX, _mouseY
,  &_buttonPar) ,_x - 110, _y);

                        _model->play_pause( keyPress( &event, Key::Space ) );
```



```
                        _model->loadSave( clickButton( &event, _mouseX, _mouseY,
  &_open) );
                        _model->saveSave( clickButton( &event, _mouseX, _mouseY,
  &_save) );
        }
        }
        return result;
}


/*#########################################################################
/#                                                                       #
/#            ACCESSEURS                                                 #
/#                                                                       #
#########################################################################*/

void GameView::setModel(GameModel* model )
{ _model = model; }

void GameView::setMenu( MenuView* menu )
{ _menu = menu; }
```

```
#ifndef GAMEVIEW_H
#define GAMEVIEW_H

/**
 * \file GameView.h
 * \brief View du jeu
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 */

#include <SFML/Graphics.hpp>
#include <string>

#include "Button.h"
#include "GameModel.h"
#include "MenuView.h"
#include "Element.h"

using namespace sf;

/**
 * \class GameView
 * \brief classe qui gÃ¨re la partie graphique du jeu
*/
class GameView : public Element
{
        private:
                RenderWindow* _window;
                GameModel* _model;
                MenuView* _menu;
                float _mouseX, _mouseY;

                Image _bgImg, _bgBar, _celluleImg, _virusImg, _selectImg;
                Sprite _bgSprite, _barSprite, _celluleSprite, _virusSprite, _sel
ectSprite;
                String _nbVirus, _nbCellule,
                                _valVirus, _valCellule;
                Clock clock;
                Button _buttonPar, _open, _save;

        public:

/// ###################### CONSTRUCTEUR & DESTRUCTEUR ##################

                GameView(const float & w, const float & h, const float & x, cons
t float & y );
                ~GameView ( );

/// ###################### ACTION FENETRE DE JEU ##################

                void draw ( );
                bool treatEvents ( );
                void updateVal();

/// ###################### ACCESSEUR ##################

                void setModel (GameModel* model );
                void setMenu ( MenuView* menu );
};

#endif // GAMEVIEW_H
```



```
/**
 * \file GrilleCellule.cc
 * \brief regroupement de Cellule
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 */

#include <iostream>
#include <cstdlib>

#include "GrilleCellule.h"
#include "enum.h"
#include "util.h"

using namespace std;

/*#########################################################################################
/#                                                                                       #
/#            CONSTRUCTEUR & DESTRUCTEUR                                                 #
/#                                                                                       #
#########################################################################################*/

/**
 * \fn GrilleCellule ( const int & mia, const int & maa, const int & mid, const
int & mad )
 * \brief Constructeur par parametre de GrilleCellule
 * \param miaa, maa, mid, mad entiers pour definit max et min voisin vivante des
 Cellules
 */
GrilleCellule::GrilleCellule ( const int & mia, const int & maa, const int & mid
, const int & mad )
: _min_alive(mia), _max_alive(maa), _min_dead(mid), _max_dead(mad)
{
        _tmpAleaCellule = _aleaCellule = 0.55;
        _tmpAleaResist = _aleaResist = 0.90;
        vector<Cellule> tmp;
        _wC = 40; _hC = 30;
        for ( int i=0; i < 16; i++ )
        {
                for ( int j=0; j < 12; j++ )
                {
                        tmp.push_back(Cellule(i*40,j*30+30, _aleaCellule, _aleaR
esist));
                }
                _grilleCellule.push_back(tmp);
                tmp.clear();
        }
}

GrilleCellule::~GrilleCellule ( )
{
}


/*#########################################################################################
/#                                                                                       #
/#            MODIFICATION MATRICE CELLULE                                                #
/#                                                                                       #
#########################################################################################*/

/**
 * \fn GrilleCellule::add_column (const float & w, const float & h )
```

```
 * \brief ajoute une colonne de Cellule
 * \param w,h rÃ©el qui donne la taille de la fenÃªtre
 * \return rien
 */
void GrilleCellule::add_column (const float & w, const float & h )
{
        int ti, tj;
        getTailleGrille(ti, tj);

        vector<Cellule> tmp;
        for ( int j=0; j<tj; j++ )
                tmp.push_back(Cellule(20,30, _aleaCellule, _aleaResist));
        _grilleCellule.push_back(tmp);
        update(w,h);

        getTailleGrille(ti, tj);
}

/**
 * \fn GrilleCellule::add_line (const float & w, const float & h )
 * \brief ajoute une ligne de Cellule
 * \param w,h rÃ©el qui donne la taille de la fenÃªtre
 * \return rien
 */
void GrilleCellule::add_line (const float & w, const float & h )
{
        int ti, tj;
        getTailleGrille(ti, tj);

        for ( int i=0; i < ti; i++ )
                _grilleCellule[i].push_back(Cellule(20,30, _aleaCellule, _aleaRe
sist));
        update(w,h);
        getTailleGrille(ti, tj);
}

/**
 * \fn GrilleCellule::remove_column (const float & w, const float & h )
 * \brief enleve une colonne de Cellule
 * \param w,h rÃ©el qui donne la taille de la fenÃªtre
 * \return rien
 */
bool GrilleCellule::remove_column (const float & w, const float & h )
{
        int ti, tj;
        getTailleGrille(ti, tj);

        if ( ti >= 1 )
        {
                _grilleCellule.pop_back();
                update(w,h);
                return 1;
        }
        else
        {
                return 0;
                cout << "impossible !" << endl;
        }

        getTailleGrille(ti, tj);
}
```



```
/**
 * \fn GrilleCellule::remove_line (const float & w, const float & h )
 * \brief enleve une ligne de Cellule
 * \param w,h rÃ©el qui donne la taille de la fenÃªtre
 * \return rien
 */
bool GrilleCellule::remove_line (const float & w, const float & h )
{
        int ti, tj;
        getTailleGrille(ti, tj);

        if ( tj >= 2 )
        {
                for ( int i=0; i < ti; i++ )
                        _grilleCellule[i].pop_back();
                update(w,h);
                return 1;
        }
        else
                return 0;

        getTailleGrille(ti, tj);
}

/**
 * \fn GrilleCellule::updateAll()
 * \brief met a jour toutes les cellules de la grille avec les nouveau parametre
 * \param rien
 * \return rien
 */
void GrilleCellule::updateAll()
{
        int ti, tj;
        getTailleGrille(ti,tj);

        for ( int i=0; i<ti; i++ )
                for ( int j=0; j<tj; j++ )
                {
// si l'utilisateur a fait des changement dans le pourcentage de vivant
                        if ( _tmpAleaCellule != _aleaCellule )
                        {
                                _grilleCellule[i][j].setEtat( aleastate(_aleaCel
lule ) );
                        }
// si l'utilisateur a fait des changement dans le pourcentage de resistant
                        if ( _tmpAleaResist != _aleaResist )
                        {
                                _grilleCellule[i][j].setSante( aleahealth(_aleaR
esist ) );
                        }
                }
        _tmpAleaResist = _aleaResist;
        _tmpAleaCellule = _aleaCellule;
}


/*#########################################################################
/#                                                                       #
/#            VIE DES CELLULES                                           #
/#                                                                       #
#########################################################################*/
```

```
/**
 * \fn GrilleCellule::update(const float & w, const float & h)
 * \brief met a jour la position de chanque cellule aprÃ©s modification de la ta
ille de la grille
 * \param w,h rÃ©el qui donne la taille de la fenÃªtre
 * \return rien
 */
void GrilleCellule::update(const float & w, const float & h)
{
        int ti, tj;
        getTailleGrille(ti, tj);
        _wC = w/ti; _hC = h/tj;
        for ( int i=0; i < ti; i++)
        {
                for ( int j=0; j < tj; j++)
                {
                        _grilleCellule[i][j].setX( i * (w/ti) );
                        _grilleCellule[i][j].setY( j * (h/tj) + 30 );
                }
        }
}

/**
 * \fn GrilleCellule::loi_de_vie()
 * \brief effectuÃ© les modifications de vie des Cellules
 * \param w,h rÃ©el qui donne la taille de la fenÃªtre
 * \return rien
 */
void GrilleCellule::loi_de_vie()
{
        int ti, tj, voisin;
        getTailleGrille(ti, tj);
        state tmp[ti][tj];

        for ( int i=0; i< ti; i++ )
        {
                for ( int j=0; j< tj; j++ )
                {
                        voisin = nb_cellule_voisine(i,j);
                        if( _grilleCellule[i][j].getEtat() == dead && voisin >=
_min_dead && voisin <= _max_dead )
                                tmp[i][j] = alive;
                        else if( _grilleCellule[i][j].getEtat() == alive && vois
in >= _min_alive && voisin <= _max_alive )
                                tmp[i][j] = alive;
                        else if( _grilleCellule[i][j].getEtat() == infest && voi
sin >= _min_dead && voisin <= _max_dead )
                                tmp[i][j] = alive;
                        else
                                tmp[i][j] = dead;
                }
        }

        for ( int i=0; i< ti; i++ )
        {
                for ( int j=0; j< tj; j++ )
                        _grilleCellule[i][j].setEtat( tmp[i][j] );
        }
}

/**
 * \fn GrilleCellule::nb_cellule_voisine(const int & a, const int & b )
```



```
 * \brief donne le nombre de voisine d'une cellule vivant
 * \param a,b entiers qui donnent l'indicide de la Cellule
 * \return entier indiquant le nombre de voisin vivant
 */
int GrilleCellule::nb_cellule_voisine(const int & a, const int & b )
{
        int ti, tj;
        int cpt=0;
        getTailleGrille(ti, tj);

        for (int i=a-1; i<=a+1; i++)
        {
                for (int j=b-1; j<=b+1; j++)
                {
                        if ( i!= a || j!= b )
                                if ( i>=0 && i<ti && j>=0 && j<tj )
                                        if ( _grilleCellule[i][j].getEtat() == a
live )
                                                cpt++;
                }
        }
        return cpt;
}


/*##############################################################################
/#                                                                             #
/#            ACCESSEURS GRILLE CELLULES                                       #
/#                                                                             #
##############################################################################*/

void GrilleCellule::getCoordonne( const int & i, const int & j, Unite & e) const
{
        e.setX( _grilleCellule[i][j].getX() );
        e.setY( _grilleCellule[i][j].getY() );
}

void GrilleCellule::getSizePicture( float & w, float & h) const
{ w = _wC; h = _hC; }

state GrilleCellule::getEtatCellule(const int & i, const int & j) const
{ return _grilleCellule[i][j].getEtat(); }

health GrilleCellule::getHealthCellule( const int & i, const int & j) const
{ return _grilleCellule[i][j].getSante(); }

void GrilleCellule::getTailleGrille( int & ti, int & tj ) const
{
        ti = _grilleCellule.size();
        tj = _grilleCellule[1].size();
}

void GrilleCellule::setEtatCellule( const int & i, const int & j, const state &
e)
{ _grilleCellule[i][j].setEtat( e ); }

void GrilleCellule::setMinAlive( const int & a )
{ _min_alive = a; }

void GrilleCellule::setMaxAlive( const int & a )
{ _max_alive = a; }
```

```
void GrilleCellule::setMinDead( const int & a )
{ _min_dead = a; }

void GrilleCellule::setMaxDead( const int & a )
{ _max_dead = a; }

void GrilleCellule::setAleaCellule( const float & a)
{ _aleaCellule = (100-a)/100; }

void GrilleCellule::setAleaResist(const float & a)
{ _aleaResist = (100-a)/100; }


/*##############################################################################
#                                                                              #
#            FLUX GRILLE CELLULES                                              #
#                                                                              #
##############################################################################*/

/**
 * \fn GrilleCellule::lireGrilleCellule( fstream & f )
 * \brief modifie les Cellules a partir d'un flux
 * \param f un flux qui contient les informations necessaire
 * \return rien
 */
void GrilleCellule::lireGrilleCellule( fstream & f )
{
        int ti, tj;
        float x,y;
        int val;
        state e;
        health s;
        vector<Cellule> tmp;

        f >> _min_alive >> _max_alive >> _min_dead >> _max_dead;
        f >> _aleaCellule >> _aleaResist;
        f >> ti >> tj;
        f >> _wC >> _hC;

        _grilleCellule.clear();
        for ( int i = 0; i < ti; i++ )
        {
                for (int j = 0; j < tj; j++)
                {
                        f >> x >> y;

                        f >> val;
                        e = getState(val);
                        f >> val;
                        s = getHealth(val);

                        tmp.push_back( Cellule(x,y,e,s) );
                }
                _grilleCellule.push_back( tmp );
                tmp.clear();
        }
}

/**
 * \fn GrilleCellule::ecrireGrilleCellule( fstream & f )
 * \brief charge les Cellules a partir d'un flux
 * \param f un flux qui contiendra les informations necessaire
```



```
 * \return rien
 */
void GrilleCellule::ecrireGrilleCellule( fstream & f )
{
        int ti, tj;
        getTailleGrille(ti, tj);

        f << _min_alive << " " << _max_alive << " " << _min_dead << " " << _max_d
ead << endl;
        f << _aleaCellule << " " << _aleaResist << endl;
        f << ti << " " << tj << endl;
        f << _wC << " " << _hC << endl;

        for ( int i = 0; i < ti; i++ )
                for (int j = 0; j < tj; j++)
                {
                        f << _grilleCellule[i][j].getX() << " " << _grilleCellul
e[i][j].getY() << endl;
                        f << int(_grilleCellule[i][j].getEtat()) << " " << int(_
grilleCellule[i][j].getSante()) << endl;
                }
}
```

```
#ifndef GRILLECELLULE_H
#define GRILLECELLULE_H

/**
 * \file GrilleCellule.h
 * \brief regroupement de Cellule
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 */

#include <vector>
#include <string>
#include <fstream>

#include "Cellule.h"
#include "Unite.h"
#include "enum.h"

using namespace std;

/**
 * \class GrilleCellule
 * \brief classe qui regroupe tous les Cellules
*/
class GrilleCellule
{
        private:
                vector< vector<Cellule> > _grilleCellule;

                int _min_alive, _max_alive;
                int _min_dead, _max_dead;
                float _wC, _hC;
                float _aleaCellule, _aleaResist, _tmpAleaCellule, _tmpAleaResist
;

        public:

/// ######################## CONSTRUCTOR & DESTRUCTOR ####################

                GrilleCellule( const int & mia, const int & maa, const int & mid
, const int & mad );
                ~GrilleCellule( );

/// ######################## MODIFICATION MATRICE CELLULE ################

                void add_column (const float & w, const float & h );
                void add_line (const float & w, const float & h );
                bool remove_column (const float & w, const float & h );
                bool remove_line (const float & w, const float & h );
                void updateAll();

/// ######################## VIE DES CELLULES ############################

                void update(const float & w, const float & h);
                void loi_de_vie ( );
                int nb_cellule_voisine(const int & a, const int & b );

/// ######################## ACCESSEUR GRILLE CELLULES ###################

                void getCoordonne( const int & i, const int & j, Unite & e) cons
t;
                void getSizePicture( float & w, float & h) const;
```



```
                state getEtatCellule(const int & i, const int & j) const;
                health getHealthCellule( const int & i, const int & j) const;
                void getTailleGrille( int & ti, int & tj ) const;

                void setEtatCellule( const int & i, const int & j, const state &
 e);
                void setMinAlive( const int & a );
                void setMaxAlive( const int & a );
                void setMinDead( const int & a );
                void setMaxDead( const int & a );
                void setAleaCellule( const float & a);
                void setAleaResist(const float & a);

/// ######################## FLUX GRILLE CELLULE ########################

                void lireGrilleCellule( fstream & f );
                void ecrireGrilleCellule( fstream & f );
};

#endif // GRILLECELLULE_H
```

```
/**
 * \file GroupeVirus.cc
 * \brief regroupement de Virus
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 */

#include <iostream>

#include  "GroupeVirus.h"

using namespace std;

/*##############################################################################
/#                                                                             #
/#            CONSTRUCTEUR & DESTRUCTEUR                                       #
/#                                                                             #
##############################################################################*/

/**
 * \fn GroupeVirus(const int & enfant, const float & vie, const int & incub )
 * \brief Constructeur parametre du GroupeVirus
 * \param enfant entier du nombre d'enfant Ã  la descendance
 * \param vie rÃ©el de durÃ©e de vie des Virus
 * \param incub rÃ©el de durÃ©e d'incubation des Virus
 */
GroupeVirus::GroupeVirus(const int & enfant, const float & vie, const int & incu
b )
: _wVirus(20), _hVirus(20), _nbEnfant(enfant), _dureVie(vie), _dureIncub(incub)
{}

GroupeVirus::~GroupeVirus ( )
{
}


/*##############################################################################
/#                                                                             #
/#            MODIFICATION GROUPE VIRUS                                        #
/#                                                                             #
##############################################################################*/

/**
 * \fn GroupeVirus::removeVirus()
 * \brief Methode qui enleve la dernier Virus ajouter
 * \param rien
 * \return rien
 */
void GroupeVirus::removeVirus()
{
        if ( _groupeVirus.size() > 0 )
        {
                _groupeVirus.pop_back();
        }

}

/**
 * \fn GroupeVirus::addVirus()
 * \brief Methode qui ajoute un Virus au Groupe
 * \param rien
 * \return rien
```



```
 */
void GroupeVirus::addVirus()
{ _groupeVirus.push_back( Virus() ); }

/**
 * \fn GroupeVirus::removeVirusAll()
 * \brief Methode qui enleve tous les Virus du Groupe
 * \param rien
 * \return rien
 */
void GroupeVirus::removeAll()
{ _groupeVirus.clear(); }

/**
 * \fn GroupeVirus::moveVirus()
 * \brief Methode qui enleve la dernier Virus ajouter
 * \param rien
 * \return rien
 */
void GroupeVirus::moveVirus()
{
        for ( int i=0; i< _groupeVirus.size(); i++ )
                _groupeVirus[i].move(_wVirus, _hVirus);
}

/**
 * \fn GroupeVirus::delVirus( const int & i )
 * \brief Methode qui retire un Virus et recalle correctement le groupe
 * \param i entier indiquant l'indice du virus a retirÃ©
 * \return rien
 */
void GroupeVirus::delVirus( const int & i )
{
        int n = getSizeGroupe();

        if ( n > 0 && i >= 0 && i < n )
        {
                for (int a = i+1; a < n; a++ )
                {
                        _groupeVirus[a-1] = _groupeVirus[a];
                }
                _groupeVirus.pop_back();
        }

}

/**
 * \fn GroupeVirus::updateVirus(const int & w, const int & h, const int & ti, co
nst int & tj)
 * \brief Methode qui met a jours la taille et la position de chaque Virus du Gr
oupe
 * \param w, h rÃ©els qui indiquant la nouvelle taille des Virus
 * \param ti, tj, entier indiquand la taille de la nouvelle GrilleCellule
 * \return rien
 */
void GroupeVirus::updateVirus(const float & w, const float & h, const int & ti,
const int & tj)
{
        int tvi = getSizeGroupe();
        float curx, cury;
        _wVirus = w;
        _hVirus = h;
```

```
        for ( int i=0; i < tvi; i++)
        {
                curx = _groupeVirus[i].getX();
                cury = _groupeVirus[i].getY();

                if ( ti != 0 ) _groupeVirus[i].setX( curx + (curx / ti ) );
                if ( tj != 0 ) _groupeVirus[i].setY( cury + (cury / tj) );
        }
}


/*#########################################################################################
/#                                                                                       #
/#            ACTION TIMES VIRUS                                                         #
/#                                                                                       #
#########################################################################################*/

/**
 * \fn GroupeVirus::lifeTimeVirus(const bool & g)
 * \brief met en place la vie des Virus dans le jeu
 * \param g boolÃ©en qui donne l'etat du jeu ( jouer ou pause )
 * \return rien
 */
void GroupeVirus::lifeTimeVirus(const bool & g)
{
        int ti = getSizeGroupe();
        float time, addtime;
        for ( int i=0; i < ti; i++ )
        {
// s'applique seulement si le virus n'a toujours pas infectÃ© une cellule
                if ( !_groupeVirus[i].getTarget() )
                {
                        time = _groupeVirus[i].getTimeLife();
                        addtime = _groupeVirus[i].getTimePassed();
// si le jeu est sur pause, on incrÃ©mente le temps passÃ©
                        if ( !g )
                                _groupeVirus[i].setTimePassed( time );
// sinon on vÃ©rifiÃ© si le temps s'est Ã©coulÃ©
                        else if( (time - addtime)  >= (_dureVie + addtime)  )
                                delVirus(i);
                }
        }
}

/**
 * \fn GroupeVirus::incubTimeVirus(const bool & g, const int & i )
 * \brief met en place la durÃ©e d'incubation des Virus dans le jeu
 * \param g boolÃ©en qui donne l'etat du jeu ( jouer ou pause )
 * \param i entier qui donne l'indice du Virus
 * \return rien
 */
void GroupeVirus::incubTimeVirus(const bool & g, const int & i )
{
        float timeInc = _groupeVirus[i].getTimeIncub();
        float addInc = _groupeVirus[i].getTimePassed();

        if ( !g )
                _groupeVirus[i].setTimeIncPassed( timeInc );
        else if ( (timeInc - addInc) >= ( _dureIncub + addInc) )
        {
                for ( int a = 0; a < _nbEnfant; a++ )
```



```
                        _groupeVirus.push_back( Virus(_groupeVirus[i]) );
                        delVirus( i );
        }
}


/*#########################################################################################
/#                                                                                       #
/#            ACCESSEURS VIRUS                                                           #
/#                                                                                       #
#########################################################################################*/

int GroupeVirus::getSizeGroupe() const
{ return _groupeVirus.size(); }

int GroupeVirus::getDureeInc() const
{ return _dureIncub; }

void GroupeVirus::getCoordonne( const int & i, float & x, float & y, float & w,
float & h ) const
{
        x = _groupeVirus[i].getX();
        y = _groupeVirus[i].getY();
        w = _wVirus;
        h = _hVirus;
}

bool GroupeVirus::getTargetVirus( const int & i) const
{ return _groupeVirus[i].getTarget(); }


void GroupeVirus::setCoordonneVirus( const int & i, const float & x, const float
 & y )
{
        _groupeVirus[i].setX( x );
        _groupeVirus[i].setY( y );
}

void GroupeVirus::setTargetVirus( const int & i, const bool & t )
{ _groupeVirus[i].setTarget(t); }

void GroupeVirus::setResetIncubVirus( const int & i )
{ _groupeVirus[i].setResetIncub(); }

void GroupeVirus::setNbEfant( const int & e )
{ _nbEnfant = e; }

void GroupeVirus::setDureVie( const float & d )
{ _dureVie = d; }

void GroupeVirus::setDureIncub( const float & d )
{ _dureIncub = d; }


/*#########################################################################################
/#                                                                                       #
/#            FLUX GROUPE VIRUS                                                          #
/#                                                                                       #
#########################################################################################*/

/**
 * \fn GroupeVirus::lireGroupeVirus( fstream & f )
```

```
 * \brief modifie les Virus a partir d'un flux
 * \param f un flux qui contient les informations necessaire
 * \return rien
 */
void GroupeVirus::lireGroupeVirus( fstream & f )
{
        int ti;
        bool t;
        float p, ip, x, y;

        f >> ti;
        f >> _wVirus >> _hVirus;
        f >> _nbEnfant >> _dureVie >> _dureIncub;

        _groupeVirus.clear();

        for( int i=0; i<ti; i++ )
        {
                f >> x >> y >> t >> p >> ip;

                _groupeVirus.push_back( Virus(x,y) );
                _groupeVirus[i].setTarget(t);
                _groupeVirus[i].setTimePassed(p);
                _groupeVirus[i].setTimeIncPassed(ip);
        }
}

/**
 * \fn GroupeVirus::lireGroupeVirus( fstream & f )
 * \brief charge les Virus a partir d'un flux
 * \param f un flux qui contiendra les informations necessaire
 * \return rien
 */
void GroupeVirus::ecrireGroupeVirus( fstream & f )
{
        int ti = getSizeGroupe();
        f << ti << endl;
        f << _wVirus << " " << _hVirus << endl;
        f << _nbEnfant << " " << _dureVie << " " << _dureIncub << endl;

        for( int i=0; i<ti; i++ )
        {
                f << _groupeVirus[i].getX() << " " << _groupeVirus[i].getY() <<
endl;
                f << _groupeVirus[i].getTarget() << " " << _groupeVirus[i].getTi
mePassed() << " " << _groupeVirus[i].getTimeIncPassed() << endl;
        }
}
```



```
#ifndef GROUPEVIRUS_H
#define GROUPEVIRUS_H

/**
 * \file GroupeVirus.h
 * \brief regroupement de Virus
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 */

#include <string>
#include <vector>
#include <fstream>

#include "Virus.h"

using namespace std;

/**
 * \class GroupeVirus
 * \brief classe qui regroupe tous les Virus
*/
class GroupeVirus
{
        private:
                float _wVirus;
                float _hVirus;
                int _nbEnfant;
                float _dureVie, _dureIncub;
                vector<Virus> _groupeVirus;

        public:

/// ######################## CONSTRUCTOR & DESTRUCTOR ####################

                GroupeVirus(const int & enfant, const float & vie, const int & i
ncub );
                ~GroupeVirus ( );

/// ######################## MODIFICATION GROUPE VIRUS ####################

                void removeVirus ( );
                void addVirus ( );
                void removeAll();
                void moveVirus();
                void delVirus( const int & i );
                void updateVirus(const float & w, const float & h, const int & t
i, const int & tj);

/// ######################## ACTION TIMES VIRUS ####################

                void lifeTimeVirus(const bool & g);
                void incubTimeVirus(const bool & g, const int & i );

/// ######################## ACCESSEURS VIRUS ####################

                int getSizeGroupe() const;
                int getDureeInc() const;
                bool getTargetVirus( const int & i) const;
                void getCoordonne( const int & i, float & x, float & y, float &
w, float & h ) const;
```

```
                void setCoordonneVirus( const int & i, const float & x, const fl
oat & y );
                void setTargetVirus( const int & i, const bool & t );
                void setResetIncubVirus( const int & i );
                void setNbEfant( const int & e );
                void setDureVie( const float & d );
                void setDureIncub( const float & d );

/// ###################### ACCESSEURS VIRUS ####################

                void lireGroupeVirus( fstream & f );
                void ecrireGroupeVirus( fstream & f );


};

#endif // GROUPEVIRUS_H
```



```
/**
 * \file image.cc
 * \brief chargement et affichage d'images
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 */


#include <iostream>

#include "image.h"
#include "enum.h"
#include "Element.h"
#include "Unite.h"
#include "Button.h"
#include "Slider.h"


using namespace std;
using namespace sf;


/*#########################################################################################
/#                                                                                       #
/#             CHARGEMENT IMAGES / TEXTES                                                #
/#                                                                                       #
#########################################################################################*/

/**
 * \fn loadimage( Image & img, Sprite & spt, string file )
 * \brief Fonction qui simplifie le chargement d'une image dans un sprite
 * \param img Images qui sera chargÃ© dans le sprite
 * \param spt Sprite qu'on doit charger
 * \param file string qui indique le chemin du fichier image
 * \return rien
 */
void loadimage( Image & img, Sprite & spt, string file )
{
        if (!img.LoadFromFile(file) )
        {
                cout << "l'images : " << file << " s'est pas chargÃ© ! " << endl;
        spt = Sprite();
        }
        else
                spt = Sprite( img );
}

/**
 * \fn loadimage( Image & img, Sprite & spt, string file, const float & x, const
 float & y, const float & w, const float & h )
 * \brief Fonction qui simplifie le chargement d'une image dans un sprite et de
Ã§a forme
 * \param img Images qui sera chargÃ© dans le sprite
 * \param spt Sprite qu'on doit charger
 * \param file string qui indique le chemin du fichier image
 * \param x, y, w, h rÃ©els qui definit taille et coordonnÃ© du sprite
 * \return rien
 */
void loadimage( Image & img, Sprite & spt, string file, const float & x, const f
loat & y, const float & w, const float & h )
{
        loadimage(img, spt, file);
```

```
        setSprite(spt, x, y, w, h);
}

/**
 * \fn loadtextefont( String & t, Font* f, const float & size, const float & x,
const float & y, const string & texte )
 * \brief Fonction qui simplifie le chargement d'un texte String
 * \param t String variable qui contiendra le texte
 * \param f Font qui correspond au type d'Ã©criture
 * \param size, x, y rÃ©els qui definit taille et coordonnÃ©e du String
 * \param texte string qui contient le texte voulu
 * \return rien
 */
void loadtextefont( String & t, Font* f, const float & size, const float & x, co
nst float & y, const string & texte )
{
        t.SetFont(*f);
        t.SetSize(size);
        t.SetPosition(x,y);
        t.SetText(texte);
        t.SetColor(Color(0,0,0));
}


/*############################################################################
#                                                                           #
#            MODIFICATION SPRITES                                           #
#                                                                           #
############################################################################*/

/**
 * \fn sizepositionimg( Sprite & spt, const float & x, const float & y, const fl
oat & w, const float & h )
 * \brief Fonction qui dÃ©finit les coordonnÃ©e et taille du sprite
 * \param spt Sprite qu'on doit modifier
 * \param x, y, w, h rÃ©els qui definit taille et coordonnÃ© du sprite
 * \return rien
 */
void sizepositionimg( Sprite & spt, const float & x, const float & y, const floa
t & w, const float & h )
{
        spt.SetPosition(x,y);
        spt.Resize(w,h);
}

/**
 * \fn setSprite( Sprite & spt, const float & x, const float & y, const float &
w, const float & h )
 * \brief Fonction qui dÃ©finit rectangle de lecture de l'image
 * \param spt Sprite qu'on doit modifier
 * \param x, y, w, h rÃ©els qui definit taille et coordonnÃ© du rectangle de lec
ture
 * \return rien
 */
void setSprite( Sprite & spt, const float & x, const float & y, const float & w,
 const float & h )
{
        Rect<int> r(x,y,x+w,y+h);
        spt.SetSubRect(r);
}
```



```
/*############################################################################
/#                                                                           #
/#            AFFICHAGE OBJETS                                               #
/#                                                                           #
############################################################################*/

/**
 * \fn afficherCellules(GameModel* model, RenderWindow* & window, Sprite* cellul
eSprite)
 * \brief Fonction qui affiche les cellules de la GrilleCellule
 * \param model GameModel du jeu
 * \param window Renderwindow fenetre de jeu qui affichera les sprites
 * \param celluleSprite Sprite Ã  afficher
 * \return rien
 */
void afficherCellules(GameModel* model, RenderWindow* & window, Sprite* celluleS
prite)
{
        int ti, tj;
        float w,h;
        model->getSizePictureCellule(w,h);
        Unite e;

        model->getTailleGrilleCellule(ti, tj);

        for ( int i=0; i < ti; i++ )
        {
                for ( int j=0; j < tj; j++ )
                {
                        model->getCoordonneCellule(i, j, e);
                        if ( model->getEtatCellule(i,j) == alive )
                        {
                                if ( model->getSanteCellule(i,j) == resist ) //
quand la cellule est resistante
                                {
                                        setSprite(*celluleSprite,0,0,247,247);
                                        sizepositionimg(*celluleSprite, e.getX()
, e.getY(), w, h);
                                        window->Draw(*celluleSprite);
                                }
                                else // si la cellule est normale
                                {
                                        setSprite(*celluleSprite,0,494,247,247);
                                        sizepositionimg(*celluleSprite, e.getX()
, e.getY(), w, h);
                                        window->Draw(*celluleSprite);
                                }
                        }
                        else if ( model->getEtatCellule(i,j) == infest )
                        {
                                setSprite(*celluleSprite,0,247,247,247);
                                sizepositionimg(*celluleSprite, e.getX(), e.getY
(), w, h);
                                window->Draw(*celluleSprite);
                        }
                }
        }
}

/**
 * \fn afficherVirus(GameModel* model, RenderWindow* & window, Sprite virusSprit
e )
```

```
 * \brief Fonction qui affiche les virus du GroupeVirus
 * \param model GameModel du jeu
 * \param window Renderwindow fenetre de jeu qui affichera les sprites
 * \param virusSprite Sprite Ã  afficher
 * \return rien
 */
void afficherVirus(GameModel* model, RenderWindow* & window, Sprite* virusSprite
 )
{
        int tg = model->getSizeGroupeVirus();
        float x, y, w, h;

        for ( int i=0; i < tg; i++ )
        {
                model->getCoordonneVirus(i, x, y, w, h);
                sizepositionimg(*virusSprite, x, y, w, h);
                window->Draw(*virusSprite);
        }
}

/**
 * \fn afficherButton( RenderWindow* & window, Button* button )
 * \brief Fonction qui affiche les objet Button
 * \param window Renderwindow fenetre de jeu qui affichera les sprites
 * \param button Button que l'on doit afficher
 * \return rien
 */
void afficherButton( RenderWindow* & window, Button* button )
{
        if( button->getHover() ) button->changeSprite(0,192,166,96);
        else if ( button->getPress() ) button->changeSprite(0,96,166,96);
        else button->changeSprite(0,0,166,96);

        window->Draw( button->getSprite() );
        window->Draw( button->getTxt() );
}

/**
 * \fn afficherSlider( RenderWindow* & window, Slider* slider )
 * \brief Fonction qui affiche les objet Slider
 * \param window Renderwindow fenetre de jeu qui affichera les sprites
 * \param slider Slider que l'on doit afficher
 * \return rien
 */
void afficherSlider( RenderWindow* & window, Slider* slider )
{
        if ( slider->getMove() ) slider->changeSprite(5,0,5,20);
        else slider->changeSprite(0,0,5,20);

        window->Draw( slider->getSpriteBg() );
        window->Draw( slider->getSpriteFg() );
        window->Draw( slider->getTxtInfo() );
        window->Draw( slider->getTxtVal() );
}

/**
 * \fn afficherselect( RenderWindow* & window, Sprite* select, const int & i, co
nst int & j, const float & w, const float & h)
 * \brief Fonction qui affiche le cadre de selection des cellules
 * \param window Renderwindow fenetre de jeu qui affichera les sprites
 * \param select Sprite contenant l'image a afficher
 * \param i, j entiers indiquant les indices d'une cellules selectionnÃ©e
```



```
 * \param w, h rÃ©els definissant la taille du selecteur
 * \return rien
 */
void afficherselect( RenderWindow* & window, Sprite* select, const int & i, cons
t int & j, const float & w, const float & h)
{
        if ( ((j-1)*h)+30 >= 30 )
        {
                sizepositionimg( *select, i*w, ((j-1)*h)+30, w, h);
                window->Draw( *select );
        }
}
```

```
#ifndef IMAGE_H
#define IMAGE_H

/**
 * \file image.h
 * \brief chargement et affichage d'images
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 */

#include <SFML/Graphics.hpp>
#include <string>

#include "GameModel.h"
#include "Button.h"
#include "Slider.h"

using namespace std;
using namespace sf;

///############## CHARGEMENT IMAGES / TEXTES ######################

void loadimage( Image & img, Sprite & spt, string file );
void loadimage( Image & img, Sprite & spt, string file, const float & x, const f
loat & y, const float & w, const float & h );
void loadtextefont( String & t, Font* f, const float & size, const float & x, co
nst float & y, const string & texte );

/// ############ MODIFICATION SPRITES #################

void sizepositionimg( Sprite & spt, const float & x, const float & y, const floa
t & w, const float & h );
void setSprite( Sprite & spt, const float & x, const float & y, const float & w,
 const float & h );

/// ############## AFFICHAGE OBJETS ####################

void afficherCellules(GameModel* model, RenderWindow* & window, Sprite* celluleS
prite);
void afficherVirus(GameModel* model, RenderWindow* & window, Sprite* virusSprite
 );
void afficherButton( RenderWindow* & window, Button* button );
void afficherSlider( RenderWindow* & window, Slider* slider );
void afficherselect( RenderWindow* & window, Sprite* select, const int & i, cons
t int & j, const float & w, const float & h);


#endif
```



```
/**
 * \file Intro.cc
 * \brief introduction du jeu
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 */

#include <iostream>

#include "Intro.h"
#include "event.h"
#include "image.h"

using namespace std;
using namespace sf;

Intro::Intro()
: _i(0), _next(654,8,138,44,0,"Passer l'intro", 12, 11, 18)
{
        if (!_start.OpenFromFile("musics/intro.ogg"))
        {
                cout << "erreur dans l'ouverture du fichier musique" << endl;
                //~ exit -1;
        }
        else
                _start.Play();

        _window = new RenderWindow(sf::VideoMode(800, 600, 32), "Game Of Life", sf
::Style::Close );  //| sf::Style::Resize
        _window->SetPosition(0,0);

        loadimage( _bgImg, _bgSprite, "images/bg.png");
        loadimage( _meduseImg, _meduseSprite, "images/meduse.png");
        loadimage( _introImg, _introSprite, "images/txtanim.png");
        loadimage( _iutImg, _iutSprite, "images/iut.png");
        loadimage( _autorImg, _autorSprite, "images/auteur.png");

        setSprite( _meduseSprite, 358, 0, 358, 298 );
        sizepositionimg( _meduseSprite, -358, 176, 358, 298 );
        setSprite( _introSprite, 0,0,0,0);
        sizepositionimg( _introSprite, 222, 336, 656, 147 );
        sizepositionimg( _bgSprite, 0, 0, 800, 600);
        sizepositionimg( _iutSprite, 7, 7, 112, 98 );
        sizepositionimg( _autorSprite, 4,559,689,34);

        _time.Reset();

}

Intro::~Intro()
{
        if( _window!= NULL )
        delete _window;
}

void Intro::draw()
{
        if ( _time.GetElapsedTime() >= 0.1 && _meduseSprite.GetPosition().x <= 8
00 )
        {
                animation();
                _time.Reset();
```

```
        }
        _window->Draw( _bgSprite );
        afficherButton(_window, &_next);
        _window->Draw( _introSprite );
        _window->Draw( _meduseSprite );
        if ( timePassedAfter(0.6) ) _window->Draw( _iutSprite );
        if ( timePassedAfter(0.9) ) _window->Draw( _autorSprite);
        usleep(40000);
        _window->Display();
}

bool Intro::treatEvents()
{
bool result = false;
        if ( timePassedAfter(9) )
        {
                result = false;
        }
        else if(_window->IsOpened())
        {
        result = true;
        sf::Event event;

                while ( _window->GetEvent(event) )
                {

                        _mouseX = _window->GetInput().GetMouseX();
                _mouseY = _window->GetInput().GetMouseY();

                if ((event.Type == Event::Closed) || keyPress( &event, Key::Esca
pe) )
                        {
                                _window->Close();
                                result = false;
                }
                if ( clickButton( &event, _mouseX, _mouseY, &_next)) result = fa
lse;

        }
        }
        return result;
}


void Intro::animation()
{
        float x = _meduseSprite.GetPosition().x;

        if ( _i == 0 )
        {
                setSprite( _meduseSprite, 0, 0, 358, 298 );
                sizepositionimg( _meduseSprite, x, 136, 358, 298 );
                _i = 1;
        }
        else if ( _i == 1 )
        {
                setSprite( _meduseSprite, 358, 0, 358, 298 );
                sizepositionimg( _meduseSprite, x+35, 136, 358, 298 );
                _i = 0;
        }
        if ( x > 84 )
        {
```



```
                setSprite( _introSprite, 0,0,(x-84),147);
                sizepositionimg( _introSprite, 30, 227, (x-30), 147 );
        }
}

bool Intro::timePassedAfter( const float & t )
{
        return ( _time.GetElapsedTime() >= t && _meduseSprite.GetPosition().x >
800 );
}
```

mai 19, 11 23:22 **Intro.h** Page 1/1

```
#ifndef INTRO_H
#define INTRO_H

/**
 * \file Intro.h
 * \brief introduction du jeu
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 */

#include <SFML/Graphics.hpp>
#include <SFML/Audio.hpp>

#include "Button.h"

using namespace sf;

/**
 * \class Intro
 * \brief classe qui gÃ¨re l'introduction
*/
class Intro
{
        private:

                RenderWindow* _window;
                float _mouseX, _mouseY;
                int _i;
                Clock _time;

                Image _bgImg, _meduseImg, _introImg, _iutImg, _autorImg;
                Sprite _bgSprite, _meduseSprite, _introSprite, _iutSprite, _auto
rSprite;
                Button _next;
                Music _start;


        public :

                Intro();
                ~Intro();
                void draw();
                bool treatEvents ();
                void animation();
                bool timePassedAfter( const float & t );
};

#endif // INTRO_H
```

mai 19, 11 23:22 **main.cc** Page 1/1

```
/**
 * \file main.cc
 * \brief main du programme
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 */

#include <iostream>

#include "GameModel.h"
#include "GameView.h"
#include "MenuView.h"
#include "util.h"
#include "Intro.h"

using namespace std;

const float SCREEN_WIDTH = 800;
const float SCREEN_HEIGHT = 600;
const float PAR_WIDTH = 198;
const float PAR_HEIGHT = 600;

int main()
{
        srand(time(NULL));

///~~ COMMENTER CETTE PARTIE POUR ENLEVER L'INTRODUCTION
// DEBUT
        Intro* introjeu = new Intro;
        while(introjeu->treatEvents())
        { introjeu->draw(); }
        delete introjeu;
// FIN

        GameModel * model = new GameModel(SCREEN_WIDTH, SCREEN_HEIGHT);

        MenuView * menuView = new MenuView(PAR_WIDTH, PAR_HEIGHT, 0, 0);
        GameView * gameView = new GameView(SCREEN_WIDTH, SCREEN_HEIGHT+30, 210,
0);

        gameView->setModel(model);
        gameView->setMenu(menuView);
        menuView->setModel(model);

        while(gameView->treatEvents())
        {
                menuView->treatEvents();
                model->nextStep();
                gameView->draw();
                menuView->draw();
        }

        delete gameView;
        delete menuView;
        delete model;

        return EXIT_SUCCESS;
}
```

```
CXX = g++
source = Element.cc Button.cc Slider.cc Unite.cc Cellule.cc Virus.cc GrilleCellu
le.cc GroupeVirus.cc image.cc main.cc util.cc event.cc enum.cc GameModel.cc Game
Save.cc GameView.cc MenuView.cc Intro.cc
LDXXFLAGS = -lsfml-audio -lsfml-graphics -lsfml-window -lsfml-system
objet = $(source:.cc=.o)

##########################################################

../jeuDeLaVie : $(objet)
        @ echo " "
        @ echo \"JEU DE LA VIE\" CREATED !
        @$(CXX) $(LDXXFLAGS) $^ -o $@

Element.o : Element.cc Element.h
        @echo - ELEMENT OBJECTS :
        @$(CXX) -c $<

Button.o : Button.cc Button.h image.h Element.h
        @echo "~~~~~~~ BUTTON"
        @$(CXX) -c $<


Slider.o : Slider.cc Slider.h Element.h image.h util.h
        @echo "~~~~~~~ SLIDER"
        @$(CXX) -c $<

Unite.o : Unite.cc Unite.h
        @ echo - UNITE OBJECTS :
        @$(CXX) -c $<

Cellule.o : Cellule.cc Cellule.h Unite.h enum.h util.h
        @echo "~~~~~~~ CELLULE"
        @$(CXX) -c  $<

Virus.o : Virus.cc Virus.h Unite.h util.h
        @echo "~~~~~~~ VIRUS"
        @$(CXX) -c $<

GrilleCellule.o : GrilleCellule.cc GrilleCellule.h Cellule.h Unite.h enum.h util.h
        @echo "~~~~~~~ GRILLE CELLULE"
        @$(CXX) -c $<

GroupeVirus.o : GroupeVirus.cc GroupeVirus.h Virus.h
        @echo "~~~~~~~ GROUPE VIRUS"
        @$(CXX) -c $<

util.o : util.cc util.h enum.h
        @ echo - UTIL FONCTIONS
        @$(CXX) -c $<

image.o : image.cc image.h enum.h Element.h Button.h Slider.h Unite.h Element.h
GameModel.h
        @echo "~~~~~~~ IMAGE"
        @$(CXX) -c $<

event.o : event.cc event.h Button.h Element.h Slider.h
        @echo "~~~~~~~ EVENT"
        @$(CXX) -c $<

enum.o : enum.cc enum.h
        @echo "~~~~~~~ ENUM"
```



```
        @$(CXX) -c $<

GameModel.o : GameModel.cc GameModel.h GameSave.h Unite.h GrilleCellule.h Groupe
Virus.h enum.h event.h
        @echo - CONTROL / VIEW OBJECTS
        @echo "~~~~~~~ GAMEMODEL"
        @$(CXX) -c $<

GameSave.o : GameSave.cc GameSave.h GrilleCellule.h GroupeVirus.h
        @echo "~~~~~~~ GAMESAVE"
        @$(CXX) -c $<

GameView.o : GameView.cc GameView.h GameModel.h MenuView.h Element.h Button.h im
age.h util.h event.h
        @echo "~~~~~~~ GAMEVIEW"
        @$(CXX) -c $<

main.o : main.cc GameModel.h GameView.h MenuView.h util.h Intro.h
        @echo "~~~~~~~ MAIN"
        @$(CXX) -c $<

MenuView.o : MenuView.cc MenuView.h image.h Element.h Button.h Slider.h GameMode
l.h event.h
        @echo "~~~~~~~ MENUVIEW"
        @$(CXX) -c $<

Intro.o : Intro.cc Intro.h Button.h event.h image.h
        @$(CXX) -c $<


########### CLEAN ###########

clean :
        @rm -f *~ *.o
        @echo -- ALL FILES .O ARE REMOVED
        @ echo " "

########### MRPROPER ###########

mrproper :
        @rm -f *~ *.o ../jeuDeLaVie
        @echo -- ALL FILES .O AND EXE ARE REMOVED
        @ echo " "

########### OPEN ###########

open :
        geany *.cc *.h Makefile &

########### DOC ############

doc :
        doxygen Doxyfile
        @cd ../doc/latex && make
        @cd ../doc/ && mv latex/refman.pdf Doc-JeuDeLaVie.pdf
```

```
/**
 * \file MenuView.cc
 * \brief View du menu
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 */

#include <iostream>

#include "MenuView.h"
#include "image.h"
#include "event.h"

using namespace std;


/*##############################################################################
/#                                                                             #
/#            CONSTRUCTEUR & DESTRUCTEUR                                       #
/#                                                                             #
##############################################################################*/

/**
 * \fn MenuView(const float & w, const float & h, const float & x, const float &
 y )
 * \brief Constructeur par parametre de MenuView
 * \param w,h des rÃ©els pour indiquÃ© la taille de la fenÃªtre
 * \param x,y des rÃ©els pour definir la position de la fenÃªtre
 */
MenuView::MenuView(const float & w, const float & h, const float & x, const floa
t & y )
: Element(x, y, w, h),

/// BOUTTON :

_playButton( 0, 475, 100, 50, 0, "jouer", 20, 8, 25),
// boutton joueur / pause

_addColum( 0, 350, 100, 50, 0, "Ajouter \nColonne", 27, 8, 15),
// boutton ajouter une colonne de cellule
_removeColum( 100, 350, 100, 50, 0, "Enlever \nColonne", 27, 8, 15),         // boutt
on enlever une colonne de cellule
_addLine( 0, 400, 100, 50, 0, "Ajouter  \nLigne", 27, 8, 15),
// boutton ajouter une ligne de cellule
_removeLine( 100, 400, 100, 50, 0, "Enlever\n  Ligne", 25, 8, 15),
// boutton enlever une ligne de cellule
_updateAll( 100, 475, 100, 50, 0,"Mise a\n Jour", 28, 8, 15),
// boutton mise a jour des cellules

_addVirus( 0, 50, 100, 50, 0, "Ajouter\n Virus", 27, 8, 15),
// boutton ajouter un virus
_removeVirus( 100, 50, 100, 50, 0, "Retirer\n Virus", 27, 8, 15),
// boutton enlever un virus
_removeAllVirus( 50, 100, 100, 50, 0, "Enlever\n Tout", 27, 8, 15),
// boutton tout les virus

/// SLIDER :

_vitesseJeu( 10, 570, 180, 10, 65, 100, 0, 18, "vitesse jeu =" ),
// vitesse du jeu

_minAliveCellule( 10, 80, 180, 10, 2, 8, 0, 18, "Min Vivant = " ),
```



```
// minimum cellule voisin pour rester vivant
_maxAliveCellule( 10, 125, 180, 10, 3, 8, 0, 18, "Max Vivant = " ),         // maxim
um cellule voisin pour rester vivant
_minDeadCellule( 10, 170, 180, 10, 3, 8, 0, 18, "Min Mort = " ),
// minimum cellule voisin pour devenir vivant
_maxDeadCellule( 10, 215, 180, 10, 3, 8, 0, 18, "Max Mort = " ),
// maximum cellule voisin pour devenir vivant
_aleaCellule( 10, 260, 180, 10, 45, 100, 0, 18, "ViVant/100 = "),
// pourcentage aleatoire pour l'apparittion de cellule vivante
_aleaResist(10, 305, 180, 10, 10, 100, 0, 18, "Resistant/100 = "),
// pourcentage aleatoire pour l'apparattion de cellule resistante

_nbEnfantVirus( 10, 195, 180, 10, 2, 8, 1, 18, "Enfant = " ),
// nombre d'enfant descendant d'un virus
_tmpLifeVirus( 10, 245, 180, 10, 6, 12, 1, 18, "Temps Vie =  " ),
// temps de vie d'un virus
_tmpIncubVirus( 10, 295, 180, 10, 2, 8, 1, 18, "Incubation = " )
// temps d'incubation d'un virus

{
        _onglet = cellule;
        _window = new RenderWindow(sf::VideoMode(_w, _h, 32), "Parametre", sf::Sty
le::Close);
        _window->SetPosition(_x,_y);

        loadimage(_bgCellImg, _bgCellSprite, "images/bg_cell.png");
        loadimage(_bgVirusImg, _bgVirusSprite, "images/bg_virus.png");
}

MenuView::~MenuView ( )
{
        if(_window != NULL) delete _window;
}


/*##############################################################################
/#                                                                             #
/#            ACTION FENETRE                                                   #
/#                                                                             #
##############################################################################*/


/**
 * \fn MenuView::draw( )
 * \brief Methode qui dessine les images et Objets du menu
 * \param rien
 * \return rien
 */
void MenuView::draw( )
{
/// SEULEMENT SI LA FENETRE EST OUVERTE
        if(_window->IsOpened())
        {
                if ( _model->getStateGame() ) _playButton.setTxt( "pause" );
                        else _playButton.setTxt( "jouer" );

                if ( _onglet == cellule ) // l'onglet cellule selectionnÃ©, affi
cher ....
                {
                        _window->Draw(_bgCellSprite);                   // fond
de menu (select cellule)
```

```
                        afficherSlider(_window, &_minAliveCellule);
// slider : min vivant Cellule
                        afficherSlider(_window, &_maxAliveCellule);
// slider : max vivant Cellule
                        afficherSlider(_window, &_minDeadCellule);
// slider : min mort Cellule
                        afficherSlider(_window, &_maxDeadCellule);
// slider : max mort Cellule
                        afficherSlider(_window, &_aleaCellule);
// slider : pourcentage Cellule vivante
                        afficherSlider(_window, &_aleaResist);
// slider : pourcentage Cellule Resistante


                }
                else // l'onglet virus selectionnÃ©, afficher ....
                {
                        _window->Draw(_bgVirusSprite);                  // fond
de menu (select virus)
                        afficherButton(_window, &_addVirus);            // boutt
on : ajouter un virus
                        afficherButton(_window, &_removeVirus);         // boutt
on : enlever un virus
                        afficherButton(_window, &_removeAllVirus);      // boutt
on : enlever tous les virus

                        afficherSlider(_window, &_nbEnfantVirus);       // slide
r : nombre d'enfant
                        afficherSlider(_window, &_tmpLifeVirus);        // slide
r : temps de vie Virus
                        afficherSlider(_window, &_tmpIncubVirus);       // slide
r : temps d'incubation
                }

                _model->setVitesse( _vitesseJeu.getVal() );
                _model->setMinAliveCellule( _minAliveCellule.getVal() );
                _model->setMaxAliveCellule( _maxAliveCellule.getVal() );
                _model->setMinDeadCellule( _minDeadCellule.getVal() );
                _model->setMaxDeadCellule( _maxDeadCellule.getVal() );
                _model->setNbEfantVirus( _nbEnfantVirus.getVal() );
                _model->setDureVieVirus( _tmpLifeVirus.getVal() );
                _model->setDureIncubVirus( _tmpIncubVirus.getVal() );

                afficherButton(_window, &_addColum);            // boutton : ajo
uter colone cellule
                afficherButton(_window, &_removeColum);         // boutton : enl
ever colone cellule
                afficherButton(_window, &_addLine);                     // boutt
on : ajouter ligne cellule
                afficherButton(_window, &_removeLine);          // boutton : enl
ever ligne cellule

                afficherButton(_window, &_playButton);                  // boutt
on : jeu en pause ou lecteur
                afficherButton(_window, &_updateAll);
                afficherSlider(_window, &_vitesseJeu);



                _window->Display();
                usleep(10000);
        }
}
```



```
/**
 * \fn MenuView::treatEvents()
 * \brief Methode qui rÃ©cupÃ©re les Ã©venements d'entrÃ©es sorties
 * \param rien
 * \return rien
 */
bool MenuView::treatEvents()
{
        bool result = false;
        if(_window->IsOpened())
        {
        result = true;
        sf::Event event;
                while ( _window->GetEvent(event) )
                {
                        // evenement permettant de fermer la fenetre
                if ((event.Type == sf::Event::Closed) ||
                        ((event.Type == sf::Event::KeyPressed) && (event.Key.Cod
e == sf::Key::Escape)))
                        {
                                _window->Close();
                                result = false;
                }

                _mouseX = _window->GetInput().GetMouseX();
                _mouseY = _window->GetInput().GetMouseY();


                // ~~~~~~~~~ AJOUTER UN VIRUS DANS LE JEU ~~~~~~~~~~~~~~~~~

                if ( _onglet == virus )
                {
                                _model->add_virus( clickButton(&event, _mouseX,
_mouseY, &_addVirus) );
                                _model->del_virus( clickButton(&event, _mouseX,
_mouseY, &_removeVirus) );
                                _model->removeAllVirus( clickButton(&event, _mou
seX, _mouseY, &_removeAllVirus) );

                                moveSlider( &event, _nbEnfantVirus.getSlider(),
&_nbEnfantVirus, _mouseX, _mouseY );
                                moveSlider( &event, _tmpLifeVirus.getSlider(), &
_tmpLifeVirus, _mouseX, _mouseY );
                                moveSlider( &event, _tmpIncubVirus.getSlider(),
&_tmpIncubVirus, _mouseX, _mouseY );
                        }

                else  // l'onglet Cellule
                {
                                moveSlider( &event, _minAliveCellule.getSlider()
, &_minAliveCellule, _mouseX, _mouseY );
                                moveSlider( &event, _maxAliveCellule.getSlider()
, &_maxAliveCellule, _mouseX, _mouseY );
                                moveSlider( &event, _maxDeadCellule.getSlider(),
 &_maxDeadCellule, _mouseX, _mouseY );
                                moveSlider( &event, _minDeadCellule.getSlider(),
 &_minDeadCellule, _mouseX, _mouseY );
                                moveSlider( &event, _aleaCellule.getSlider(), &_
aleaCellule, _mouseX, _mouseY );
                                moveSlider( &event, _aleaResist.getSlider(), &_a
leaResist, _mouseX, _mouseY );
```

```
                        }

                        updateSlider();

                moveSlider( &event, _vitesseJeu.getSlider(), &_vitesseJeu, _mous
eX, _mouseY );

                _model->add_line( clickButton(&event, _mouseX, _mouseY, &_addLin
e) );
                        _model->remove_line( clickButton(&event, _mouseX, _mouse
Y, &_removeLine) );
                        _model->add_column( clickButton(&event, _mouseX, _mouseY
, &_addColum) );
                        _model->remove_column( clickButton(&event, _mouseX, _mou
seY, &_removeColum) );

                _model->play_pause( clickButton(&event, _mouseX, _mouseY, &_play
Button) );
                _model->update_all_Cellule( clickButton(&event, _mouseX, _mouseY
, &_updateAll) );


                changeTabCellule( (event.Type == sf::Event::MouseButtonPressed)
                                                        && _mouseX >= 0 && _m
ouseX <=100 && _mouseY >= 0 && _mouseY <= 40 );
                        changeTabVirus( (event.Type == sf::Event::MouseButtonPre
ssed)
                                                        && _mouseX >= 100 &&
_mouseY >= 0 && _mouseY <= 40 );

        }
        }
        return result;
}

/**
 * \fn MenuView::updateSlider()
 * \brief Methode qui met a jours les valeurs de chaque slider du Menu
 * \param rien
 * \return rien
 */
void MenuView::updateSlider()
{
        _model->setVitesse( _vitesseJeu.getVal() );
        _model->setMinAliveCellule( _minAliveCellule.getVal() );
        _model->setMaxAliveCellule( _maxAliveCellule.getVal() );
        _model->setMinDeadCellule( _minDeadCellule.getVal() );
        _model->setMaxDeadCellule( _maxDeadCellule.getVal() );
        _model->setAleaCelluleGrille( _aleaCellule.getVal() );
        _model->setAleaResistGrille( _aleaResist.getVal() );

        _model->setNbEfantVirus( _nbEnfantVirus.getVal() );
        _model->setDureVieVirus( _tmpLifeVirus.getVal() );
        _model->setDureIncubVirus( _tmpIncubVirus.getVal() );
}

/**
 * \fn MenuView::openWindow(const bool & open, int x, int y)
 * \brief Methode qui reouvre la fenetre du MenuView
 * \param open boolÃ©en venant du Gameview aprÃ©s un click sur un boutton
 * \param x, y entiers qui definissent la position d'apparition de la fenetre me
nu
```



```
 * \return rien
 */
void MenuView::openWindow(const bool & open, int x, int y)
{
        if ( ! getWindow() && open )
        {
                delete _window;
                _window = new RenderWindow(sf::VideoMode(_w, _h, 32), "Parametre G
O F", sf::Style::Close);
                //~ _window->SetPosition(0,0);
                //~ _window->Display();
        }


}


/*############################################################################
/#                                                                           #
/#            CHANGEMENT ONGLETS                                             #
/#                                                                           #
############################################################################*/

/**
 * \fn MenuView::changeTabCellule( const bool & t )
 * \brief mÃ©thode qui ouvre l'onglet "Cellule"
 * \param t boolÃ©en qui indique si on a cliquÃ© dans la zone de l'onglet
 * \return rien
 */
void MenuView::changeTabCellule( const bool & t )
{
        if ( t )
        {
                if ( _onglet == virus )
                {
                        _onglet = cellule;
                }
        }
}

/**
 * \fn MenuView::changeTabVirus( const bool & t )
 * \brief mÃ©thode qui ouvre l'onglet "Virus"
 * \param t boolÃ©en qui indique si on a cliquÃ© dans la zone de l'onglet
 * \return rien
 */
void MenuView::changeTabVirus( const bool & t )
{
        if ( t )
        {
                if ( _onglet == cellule )
                {
                        _onglet = virus;
                }
        }
}

/*############################################################################
/#                                                                           #
/#            ACCESSEURS MENU                                                #
/#                                                                           #
############################################################################*/
```

```
bool MenuView::getWindow() const
{ return _window->IsOpened(); }

void MenuView::setModel (GameModel * model )
{
        _model = model;
}
```



```
/**
 * \file MenuView.h
 * \brief View du menu
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 */

#ifndef MENUVIEW_H
#define MENUVIEW_H

#include <SFML/Graphics.hpp>
#include <string>

#include "Button.h"
#include "Slider.h"
#include "GameModel.h"
#include "Element.h"

using namespace sf;

/**
 * \class MenuView
 * \brief classe qui gÃ¨re la partie graphique du jeu
*/
class MenuView : public Element
{
///################ ATRIBUTS ####################
        private:
                GameModel * _model;
                RenderWindow* _window;
                tab _onglet;
                int _mouseX, _mouseY;

        // ~~~~~~~~~~~~~ IMAGES & SPRITES ~~~~~~~~~~~~~

                Image _bgCellImg, _bgVirusImg;
                Sprite _bgCellSprite, _bgVirusSprite;

        // ~~~~~~~~~~~~~ BOUTTON ~~~~~~~~~~~~~

                Button  _playButton,
                                _addColum, _removeColum, _addLine, _removeLine,
                                _addVirus, _removeVirus, _removeAllVirus, _updat
eAll;

        // ~~~~~~~~~~~~~ SLIDER ~~~~~~~~~~~~~

                Slider  _vitesseJeu,
                                _minAliveCellule, _maxAliveCellule, _minDeadCell
ule, _maxDeadCellule, _aleaCellule, _aleaResist,
                                _nbEnfantVirus, _tmpLifeVirus, _tmpIncubVirus;

        public:

/// ################## CONSTRUCTEUR DESTRUCTEUR ###############

                MenuView(const float & w, const float & h, const float & x, cons
t float & y );
                ~MenuView ( );

/// ################## ACTION FENETRE #################
```

```
                void draw ( );
                bool treatEvents ( );
                void updateSlider();
                void openWindow(const bool & open, int x, int y);

/// ################## CHANGEMENT ONGLETS #################

                void changeTabCellule( const bool & t );
                void changeTabVirus( const bool & t );

/// ################## ACCESSEURS MENU #################

                bool getWindow() const;
                void setModel (GameModel * model );
                //~ void setValVitesse( const int & v );
};

#endif // MENUVIEW_H
```



```
/**
 * \file Slider.cc
 * \brief molette de reglage avec texte
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 */

#include <iostream>

#include "Slider.h"
#include "image.h"
#include "util.h"

using namespace std;
using namespace sf;


/*##############################################################################
/#                                                                             #
/#            CONSTRUCTEUR & DESTRUCTEUR                                       #
/#                                                                             #
##############################################################################*/

/**
 * \fn Slider(const int & bx, const int & by, const int & bw, const int & bh,
                            const int & val, const int & max, const int & min,
                            const int & tt, const string & txt )
 * \brief Constructeur par parametre du Slider
 * \param bx, by, bw, bh entiers qui definissent les coordonnÃ©es et taille du f
ond du Slider
 * \param val, max, min entiers qui definisse la valeur de depart, la valeur max
, et la valeur min du Slider
 * \param tt rÃ©el definisant la taille du texte
 * \param txt string contenant le texte qui va avec le Slider
 */
Slider::Slider(const int & bx, const int & by, const int & bw, const int & bh,
                            const int & val, const int & max, const int & min,
                            const int & tt, const string & txt )
: _posSlider(bx, by-(bh/2), bw*5/120, 2*bh),
Element(bx,by,bw,bh),
_val_max(max), _val_min(min), _val(val), _move(false)
{
        _xMax = bx+bw-_posSlider.getW();
        _coef = ( max - min ) / (_xMax - bx );
        _ord = max - ( _coef * _xMax );
        _posSlider.setX( (_val - _ord) / _coef );

        loadimage(_slider_fg_img, _slider_fg_sprite, "images/slider_fg.png" );
        sizepositionimg(_slider_fg_sprite, _posSlider.getX(), _posSlider.getY(),
 _posSlider.getW(), _posSlider.getH());
        loadimage(_slider_bg_img, _slider_bg_sprite, "images/slider_bg.png" );
        sizepositionimg(_slider_bg_sprite, bx, by, bw, bh);

        _font = new Font();
        _font->LoadFromFile("images/antique.ttf");

        loadtextefont( _txt_info, _font, tt, bx, by - ( tt + 10 ), txt );
        loadtextefont( _txt_val, _font, tt, txt.length() * (tt/2) + bx, by - ( t
t + 10 ), toString(_val) );
}

Slider::~Slider ( )
```

```
{
        delete _font;
}


/*#############################################################################
/#                                                                           #
/#            ACTION SLIDER                                                  #
/#                                                                           #
#############################################################################*/

/**
 * \fn Slider::updateSlider()
 * \brief Methode qui met a jours la position et la valeur du slider
 * \param rien
 * \return rien
 */
void Slider::updateSlider()
{
        if ( _posSlider.getX() > _xMax ) _posSlider.setX( _xMax );
        if ( _posSlider.getX() <= getX() ) _posSlider.setX( getX() );
        _val = (_coef * _posSlider.getX()) + _ord;
        _posSlider.setX( (_val - _ord) / _coef );
        sizepositionimg(_slider_fg_sprite, _posSlider.getX(), _posSlider.getY(),
 _posSlider.getW(), _posSlider.getH());
        _txt_val.SetText(toString(_val) );

}

/**
 * \fn Slider::updateSlider()
 * \brief Methode qui met a jours de la valeur du slider
 * \param rien
 * \return rien
 */
void Slider::updateSliderLoad()
{
        if ( _posSlider.getX() > _xMax ) _posSlider.setX( _xMax );
        if ( _posSlider.getX() <= getX() ) _posSlider.setX( getX() );
        _posSlider.setX( (_val - _ord) / _coef );
        sizepositionimg(_slider_fg_sprite, _posSlider.getX(), _posSlider.getY(),
 _posSlider.getW(), _posSlider.getH());
        _txt_val.SetText(toString(_val) );

}

/**
 * \fn Slider::changeSprite(const float & x, const float & y, const float & w, c
onst float & h)
 * \brief change le rectangle de lecture de l'image du Sprite
 * \param x,y,w,h rÃ©els definissant la position et la taille du rectangle de le
cture
 * \return rien
 */
void Slider::changeSprite(const float & x, const float & y, const float & w, con
st float & h)
{
        setSprite(_slider_fg_sprite,x,y,w,h);
        sizepositionimg(_slider_fg_sprite, _posSlider.getX(), _posSlider.getY(),
 _posSlider.getW(), _posSlider.getH() );
}
```



```
/*#############################################################################
/#                                                                           #
/#            ACCESSEURS SLIDER                                              #
/#                                                                           #
#############################################################################*/

Sprite Slider::getSpriteBg() const
{ return _slider_bg_sprite; }

Sprite Slider::getSpriteFg() const
{ return _slider_fg_sprite; }

String Slider::getTxtInfo() const
{ return _txt_info; }

String Slider::getTxtVal() const
{ return _txt_val; }

int Slider::getVal() const
{ return _val; }

Element* Slider::getSlider()
{ return &_posSlider; }

bool Slider::getMove() const
{ return _move; }

bool Slider::setMove( const bool & m )
{ _move = m; }

void Slider::setVal( const int & v )
{
        _val = v;
        updateSliderLoad();
}
```

```
#ifndef SLIDER_H
#define SLIDER_H

/**
 * \file Slider.h
 * \brief molette de reglage avec texte
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 */

#include <SFML/Graphics.hpp>
#include <string>

#include "Element.h"

using namespace std;
using namespace sf;

/**
 * \class Slider
 * \brief classe qui gÃ¨re une molette de rÃ©glage
*/
class Slider : public Element
{
        private:
                string _nom_slider;
                Sprite _slider_fg_sprite, _slider_bg_sprite;
                String _txt_info, _txt_val;
                int _val_max, _val_min, _val;
                float _xMax, _coef, _ord;
                bool _move;
                Element _posSlider;
                Image _slider_fg_img, _slider_bg_img;
                Font * _font;


        public:

/// ###################### CONSTRUCTOR & DESTRUCTOR ###################

                Slider(const int & bx, const int & by, const int & bw, const int
 & bh,
                                const int & val, const int & max, const int & mi
n,
                                const int & tt, const string & txt );
                ~Slider( );

/// ###################### ACTION SLIDER ###################

                void updateSlider();
                void updateSliderLoad();
                void changeSprite(const float & x, const float & y, const float
& w, const float & h);

/// ###################### ACCESSEURS SLIDER ###################

                Sprite getSpriteBg() const;
                Sprite getSpriteFg() const;
                String getTxtInfo() const;
                String getTxtVal() const;
                Element* getSlider();
                int getVal() const;
```



```
                bool getMove() const;

                bool setMove( const bool & m );
                void setVal( const int & v );

};

#endif // SLIDER_H
```

mai 19, 11 23:22 **Unite.cc** Page 1/1

```cpp
/**
 * \file Unite.cc
 * \brief unitÃ© du jeu
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 */

#include "Unite.h"

/*#########################################################################################
/#                                                                                       #
/#            CONSTRUCTEUR & DESTRUCTEUR                                                 #
/#                                                                                       #
#########################################################################################*/

Unite::Unite(const float & x, const float & y )
: _x(x), _y(y)
{}

Unite::Unite()
{}

Unite::~Unite ( )
{}


/*#########################################################################################
/#                                                                                       #
/#            ACCESSEURS LECTURE                                                         #
/#                                                                                       #
#########################################################################################*/

float Unite::getX ( ) const
{ return _x; }

float Unite::getY ( ) const
{ return _y; }


/*#########################################################################################
/#                                                                                       #
/#            ACCESSEURS ECRITURE                                                        #
/#                                                                                       #
#########################################################################################*/

void Unite::setX (const float & x )
{ _x = x; }

void Unite::setY (const float & y )
{ _y = y; }
```

mai 19, 11 23:22 **Unite.h** Page 1/1

```cpp
#ifndef UNITE_H
#define UNITE_H

/**
 * \file Unite.h
 * \brief unitÃ© du jeu
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 */

/**
 * \class Unite.h
 * \brief classe mÃ¨re qui gÃ¨re une unitÃ©
*/
class Unite
{
        protected:

                float _x;
                float _y;

        public:

/// ######################### CONSTRUCTEUR DESTRUCTEUR ###################

                Unite(const float & x, const float & y);
                Unite();
                ~Unite();

/// ######################### ACCESSEUR LECTURE ##########################

                float getX ( ) const;
                float getY ( ) const;

/// ######################### ACCESSEUR ECRITURE #########################

                void setX (const float & x );
                void setY (const float & y );
};

#endif // UNITE_H
```

```
/**
 * \file util.cc
 * \brief fonctions utiles
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 */

#include <iostream>
#include <sstream>

#include "util.h"


using namespace std;
using namespace sf;


float aleafloat(float a, float b)
{
        return ( rand()/(float)RAND_MAX ) * (b-(a)) + a ;
}

int aleaint(int a, int b)
{
        return Randomizer::Random(a,b) ;
}

int aleadirect( const int & a, const int & b )
{
        if ( Randomizer::Random(a,b) >= 0 )
                return b;
        else
                return a;
}


bool aleabool(float pct)
{
        if ( aleafloat(0,1) > pct )
                return true;
        else
                return false;
}

state aleastate(float pct)
{
        if ( aleabool(pct) )
                return alive;
        else
                return dead;
}

health aleahealth(float pct)
{
        if ( aleabool(pct) )
                return resist;
        else
                return normal;
}

string toString( const int & e )
{
```



```
        ostringstream s;
        s << e;
        return s.str();
}
```

```
#ifndef UTIL_H
#define UTIL_H

/**
 * \file util.h
 * \brief fonctions utiles
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 */

#include <SFML/Graphics.hpp>
#include <iostream>
#include <string>

#include "enum.h"

using namespace std;
using namespace sf;


float aleafloat(float a, float b);

int aleaint(int a, int b);

int aleadirect( const int & a, const int & b );

bool aleabool(float pct);

state aleastate(float pct);

health aleahealth(float pct);

string toString( const int & e );

#endif
```



```
/**
 * \file Virus.cc
 * \brief unitÃ© perturbateur du jeu
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 */

#include <iostream>

#include "Virus.h"
#include "util.h"

using namespace std;
using namespace sf;


/*#########################################################################################
#                                                                                        #
#            CONSTRUCTEUR & DESTRUCTEUR                                                  #
#                                                                                        #
#########################################################################################*/

Virus::Virus()
: Unite( aleaint(10,790) , aleaint(40, 590)),
 _target(false), _dx(aleaint(-2,2)), _dy(aleaint(-2,2))
{
        _timeInc = _timePassed = 0;
        _timeDir = 0;
        _timeLife.Reset();
}

/**
 * \fn Virus( const int & x, const int & y )
 * \brief Constructeur par parametre de Virus
 * \param x,y rÃ©els definissant la position d'un Virus
 */
Virus::Virus( const float & x, const float & y )
: Unite( x, y), _target(false)
{
        _timePassed = 0;
        _timeDir = 0;
        _timeLife.Reset();
}

/**
 * \fn Virus( const Virus & v )
 * \brief Constructeur par copie de Virus
 * \param v Virus Ã  copier
 */
Virus::Virus( const Virus & v )
: Unite(v.getX(),v.getY())
{
        _timePassed = 0;
        _timeDir = 0;
        _timeLife.Reset();
}

Virus::~Virus( )
{ }


/*#########################################################################################
```

```
/#                                                                          #
/#            MOUVEMENTS VIRUS                                              #
/#                                                                          #
############################################################################*/

/**
 * \fn Virus::move(const int & w, const int & h )
 * \brief Methode qui definit le mouvement d'une cellule Ã  la prochaine etape
 * \param w, h rÃ©els qui donnent la taille du virus qui permettra le deplacemen
t
 * \return rien
 */
void Virus::move(const float & w, const float & h )
{
        if ( !_target  )
        {
                _timeDir++;
                if ( _timeDir >= 5 )
                {
                        _dx = aleadirect(-1, 1);
                        _dy = aleadirect(-1, 1);
                        _timeDir = 0;
                }
                if ( getX() <= 0 )
                {
                        setX( 1 );
                        _dx = _dx * -1;
                }
                else if( (getX()-w) >= 798 )
                {
                        setX( 799 );
                        _dx = _dx * -1;
                }
                if ( getY() <= 30 )
                {
                                setY(31);
                         _dy = _dy * -1;
                }
                else if ( (getY()-h) >= 628 )
                {
                        setY(629);
                         _dy = _dy * -1;
                }

                _x = _x + ( _dx*w );
                _y = _y + ( _dy*h  );
        }
}


/*############################################################################
/#                                                                          #
/#            ACCESSEURS VIRUS                                              #
/#                                                                          #
############################################################################*/

bool Virus::getTarget() const
{ return _target; }

float Virus::getTimeLife() const
{ return _timeLife.GetElapsedTime(); }
```



```
float Virus::getTimeIncub() const
{ return _timeIncub.GetElapsedTime(); }

float Virus::getTimePassed() const
{ return _timePassed; }

float Virus::getTimeIncPassed() const
{ return _timeInc; }


void Virus::setTimePassed( const float & t )
{ _timePassed = t; }

void Virus::setTimeIncPassed( const float & t )
{ _timeInc = t; }

void Virus::setResetIncub()
{ _timeIncub.Reset(); }

void Virus::setResetLife()
{ _timeLife.Reset(); }

void Virus::setTarget( const bool & t )
{ _target = t; }
```

```
#ifndef VIRUS_H
#define VIRUS_H

/**
 * \file Virus.h
 * \brief unitÃ© perturbateur du jeu
 * \author Mayira.Y     Arnaud.L
 * \date avril/mai 2011
 */

#include <SFML/Graphics.hpp>

#include "Unite.h"

using namespace sf;

/**
 * \class Virus
 * \brief classe qui gÃ¨re un Virus
*/
class Virus : public Unite
{
        private:
                bool _target;
                float _dy;
                float _dx;
                float _timeDir;
                Clock _timeLife, _timeIncub;
                float _timePassed, _timeInc;

        public:

/// ###################### CONSTRUCTOR & DESTRUCTOR ###################

                Virus();
                Virus( const float & x, const float & y );
                Virus( const Virus & v );
                ~Virus( );

/// ###################### MOUVEMENT VIRUS ###################

                void move(const float & w, const float & h );

/// ###################### ACCESSEURS VIRUS ###################

                bool getTarget() const;
                float getTimeLife() const;
                float getTimeIncub() const;
                float getTimePassed() const;
                float getTimeIncPassed() const;

                void setTarget( const bool & t );
                void setResetLife();
                void setResetIncub();
                void setTimePassed( const float & t );
                void setTimeIncPassed( const float & t );
};

#endif // VIRUS_H
```