Projet RedSquare

0-Mandat

Créé un jeu ou on doit tenir le plus longtemps possible sans se faire frapper par les formes en mouvement.

1-Analyse lexicale

|  |  |  |
| --- | --- | --- |
| **explicite**  **Verbe,action**  **-stratégiquement positionné**  **-empecher**  **-toucher**  **-rebondire**  réinitialiser  sauvegarder  requérir  effacer | **Nom**  -cadre  -rectangle  - **carre**  **-pion**  -pointeur  -souris  Vitesse  Diagonal  Widget  Fonctionalité  Session  Parties  Menu  le nom d'un joueur  score  niveau de difficulté  déplacements  angles  souris enfoncée  objets | **Adj/attribut**  -noir  -bleu  **-blanc**  **-rouge**  -constante  Extérieur  déroulant |
| **Implicite**  -enregistrer score  -afficher score |  |  |
| **Supplémentaire**  - | Powerups  Self.rebondir  Modifier grosseur aire de jeu  Overlap  ralentir les pions  rendre invisible aux pions |  |

2- Cas d'usage

Demarrer jeu

Cliquer sur le carré rouge

Déplacer le carré rouge

3-Scénario d'utilisation

Demarrer programme:

|  |  |
| --- | --- |
| Humain  Clique sur le programme | Machine  Créé objet contoleur  Ctr crée objet modele et vue  Vue.init |

Interface menu

|  |  |
| --- | --- |
| Clique sur un choix du menu | Controleur gère le choix avec la méthode menu(eventCode) |
|  | Initialise et lance l'option choisie |

Interface High score (lorsque choisi a partir du menu)

|  |  |
| --- | --- |
|  | Montre un menu déroulant avec les scores et les noms des personnes qui ont joué en utilisant la fonction du controleur afficherHighScore() |

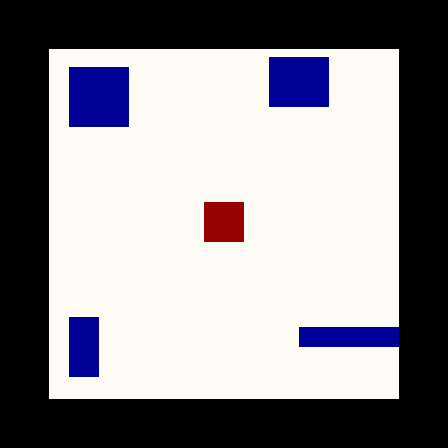
Interface Jeu

|  |  |
| --- | --- |
| Tant que le joueur ne clique pas sur le carré rouge | Le jeu est immobile. |
| Clique sur le carré rouge | Les carrés bleus commencent à bouger dans une direction aléatoire |
|  | Si les carrés bleu touche un bord ils rebondissent |
| Le joueur déplace le carré rouge en gardant la souris enfoncé | La position du carré rouge est mise à jour |
| Le joueur touche un bord ou un carré bleu | Fin de partie: Interface fin de jeu |

Interface fin de jeu

|  |  |
| --- | --- |
| Le joueur vient de perdre | La vue sort un text box pour que le joueur entre son nom |
| Le joueur valide (entre son nom) | La vue appelle setHighScore(unNom) |
|  | La fonction setHighScore(unNom) appelle à son tour sa fonction équivalent du modèle |

4- Maquette



5- Planification générale

* Class Modèle
  + \_\_init\_\_(self, grandeurEspaceDeJeuX =600, grandeurEspaceDeJeuY=600)
    - Initialise le modèle
    - Permet de changer la grandeur de l'espace de jeu
    - Lance l'initialisation de la liste de forme
  + mettreA\_Jour(self,x,y)
    - Change position des formes(joueur et ordinateur)
  + initForme()
    - Initialise la liste de forme
  + getHighScore()
    - Retourne la liste des highscores
  + setHighScore(name, temps)
    - Ecrit les highscores dans un fichier texte
* Class Forme
  + \_\_init\_\_(x1,y1,x2,y2,couleur,tag)
    - Initialise les formes du jeux
  + estEnCollision(self, nbCoin, listePointX, listePointY, x, y)
    - Vérifie si une forme en mouvement est en collision avec une autre forme
  + checkCollision(self, listeForme)
    - Itération sur la liste de forme pour vérifier sur chacun une forme est en collision avec le reste
* Class Manuel(Forme)
  + \_\_init\_\_(self, x1, y1, x2, y2, couleur, tag)
    - Initialise une forme manuelle
  + isNotDead( self, listeForme )
    - Vérifie si la forme "joueur" est morte
* Class Automatique(Forme)
  + \_\_init\_\_(self, x1, y1, x2, y2, couleur, tag)
    - Initialise une forme automatique
  + initModDir( self )
    - initialise la modification de direction des formes automatique
  + rebondire( self, forme )
    - change la direction en fonction de leur direction actuelle
  + deplacer( self, listeForme )
    - Gère le déplacement de la Forme
  + changerPosition( self )
    - Change la position de la Forme selon son modificateur
  + toucheUnBord(self, listeForme)
    - Verifie si la Forme touche un bord
* Class Vue
  + \_\_init\_\_(self, parent)
    - Initialise la vue
    - Créer le Canvas
    - Créer les binds
  + gotbouge(self,event)
    - Donne la position X et Y du joueur
  + gotclick(self,event)
    - Vérifie si le click est sur la forme du joueur
  + forgotclick(self,event)
    - Si le click est laché, la position du joueur n'est plus updaté
  + menu( self )
    - Créer un menu
  + menuHighScore( self ):
    - Créer un menu highscore
  + mettreA\_Jour( self, listeForme)
    - Met à jour le canevas
    - Dessine la surface de jeu
  + getPositionCarre( self )
    - Retourne la position du joueur
  + afficherFinDePartie(self, score)
    - affiche un text box pour rentrer le nom du joueur
    - Retourne le nom du joueur
* Class Contrôleur
  + \_\_init\_\_(self)
    - Initialise le controleur
    - Lance l'initialisation de la vue
    - Créer une variable pour le modèle sans l'initialiser
  + pause(self)
    - Pause le jeu
  + menu(self, eventCode)
    - Gère les options du menu
  + gameLoop(self)
    - Fait rouler le jeu
  + afficherHighScore(self)
    - lance le menu highscore
  + setHighScore(self,unNom)
    - Set le highscore

6- Modèle de donnée

|  |  |
| --- | --- |
| Modele  Propriétaire : Xavier, Alexandre | Controleur, Forme |
| Fonctions:   * \_\_init\_\_(self, grandeurEspaceDeJeuX =600, grandeurEspaceDeJeuY=600) * mettreA\_Jour(self,x,y) * initForme() * getHighScore() * setHighScore(name)   Variables:  gEDJX  gEDJY  listeForme[]  nbForme |

|  |  |
| --- | --- |
| Forme(object)  Propriétaire : Xavier, Alexandre | Modele, Controleur |
| Fonctions:   * \_\_init\_\_(x1,y1,x2,y2,couleur,tag) * estEnCollision(self, nbCoin, listePointX, listePointY, x, y) * checkCollision(self, listeForme)   Variables:   * x1 * x2 * y1 * y2 |  |

|  |  |
| --- | --- |
| Manuel(Forme)  Propriétaire : Xavier, Alexandre | Modele, Controleur |
| Fonctions:  \_\_init\_\_(self, x1, y1, x2, y2, couleur, tag)  isNotDead( self, listeForme ) |  |

|  |  |
| --- | --- |
| Automatique(Forme)  Propriétaire : Xavier, Alexandre | Modele, Controleur |
| Fonctions:   * \_\_init\_\_(self, x1, y1, x2, y2, couleur, tag) * initModDir( self ) * rebondire( self, forme ) * deplacer( self, listeForme ) * changerPosition( self ) * toucheUnBord(self, listeForme) |  |

|  |  |
| --- | --- |
| Vue  Propriétaire : Xavier, Alexandre | Controleur |
| Fonctions:   * \_\_init\_\_(self, parent) * gotbouge(self,event) * gotclick(self,event) * forgotclick(self,event) * menu( self ) * menuHighScore( self ): * mettreA\_Jour( self, listeForme) * getPositionCarre( self )   Variables:  self.parent = parent  self.root = Tk()  self.carrebouge = 0  self.canevas = Canvas(self.root,width=600,height=600,bg="white")  self.posCarreX = None  self.posCarreY = None  self.premierClick = 1 |

|  |  |
| --- | --- |
| Controleur  Propriétaire : Xavier, Alexandre | Modele, Vue, Forme |
| Fonctions:   * \_\_init\_\_(self) * pause(self) * menu(self, eventCode) * gameLoop(self) * afficherHighScore(self) * setHighScore(self,unNom)   Variables:  self.compteurDeLoop = 0  self.isRunning = 0  self.modele = None  self.vue = fichierVue.Vue( self ) |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Liste de situations et de fonctions du jeu** | |  |  |  |
| Cette section sert à préciser les scénarii qui seront développés lors des sprints: vous devez également les ordonner afin de produire les choses plus importantes le plus vite possible (afin notamment de réaligner le tir au besoin) | | | | |
|
|
|
| LEGENDE : A = tres important, B = important, C= normal, D=secondaire, E = supplementaire | | | | |
| **Fonctionnalités** | **Priorité** | **Prévision en heures** | **Sprint visé** | **% de réalisation** |
| ***Vue()*** | **A** | **126mins** |  | **70%** |
| *\_\_Init\_\_()* | A | 25min | 2 | 75% |
| Gotbouge() | A | 5mins |  | 100%% |
| Gotclick() | A | 5mins |  | 100% |
| Forgotclick() | A | 5mins |  | 100% |
| Menu() | C | 25mins | 3 | 25% |
| menuHighScore() | E | 25mins | 3 | 25% |
| mettreA\_Jour(listeForme, temps)  \*\*temps afficher dans le coin = supp | A | 5mins | 2 | 75% |
| *initPartie(listeForme)* | A | 20mins | 2 | 0% |
| *getPositionCarre()* | A | 1mins |  | 100% |
| menuSetHighScore() | E | 10mins | 3 | 0% |
|  |  |  |  |  |
| ***Modele()*** |  | **65mins** |  | **100%** |
| *\_\_init\_\_()* | A | 10mins |  | 100% |
| mettreA\_jour(x,y) | A | 5mins |  | 100% |
| initForme() | A | 30mins |  | 100% |
| getHighScore() | E | 10mins |  | 100% |
| *setHighScore(name, temps)* | E | 10mins |  | 100% |
|  |  |  |  |  |
| **Controleur()** | **A** | **54mins** |  | **82%** |
| \_init\_\_() | A | 10mins |  | 100% |
| Pause() | E | 2mins |  | 100% |
| Menu() | C | 10mins |  | 100% |
| *gameLoop()* | A | 30mins | 2 | 75% |
| *afficherHighScore()* | E | 1mins |  | 100% |
| *setHighScore(unNom)* | E | 1mins |  | 100% |
| *getDiffTemps* | E | 5mins | 3 | 0% |
|  |  |  |  |  |
| ***Forme()*** | **A** | **60mins** |  | **100%** |
| *\_\_init\_\_(x1,y1,x2,y2, couleur,tag)* | B | 25mins |  | 100% |
| *estEnCollision(nbCoin, listePointX,listePointY,x,y)* | B | 25mins |  | 100% |
| *checkCollision(listeForme)* | B | 10mins |  | 100% |
|  |  |  |  |  |
| ***Manuel(Forme)*** | **B** | **6mins** |  | **100%** |
| *\_\_init\_\_(x1,y1,x2,y2,couleur,tag)* | A | 1mins |  | 100% |
| *isNotDead(listeForme)* | B | 5mins |  | 100% |
|  |  |  |  |  |
| ***Automatique(Forme)*** | **A** | **40mins** |  | **100%** |
| *\_\_init\_\_(x1,y1,x2,y2,couleur,tag)* | A | 10mins |  | 100% |
| *initModDir()* | A | 5mins |  | 100% |
| *Rebondire(forme)* | A | 10mins |  | 100% |
| *Deplacer(listeForme)* | A | 10mins |  | 100% |
| *changerPosition()* | A | 5mins |  | 100% |