A vos Risk et périls

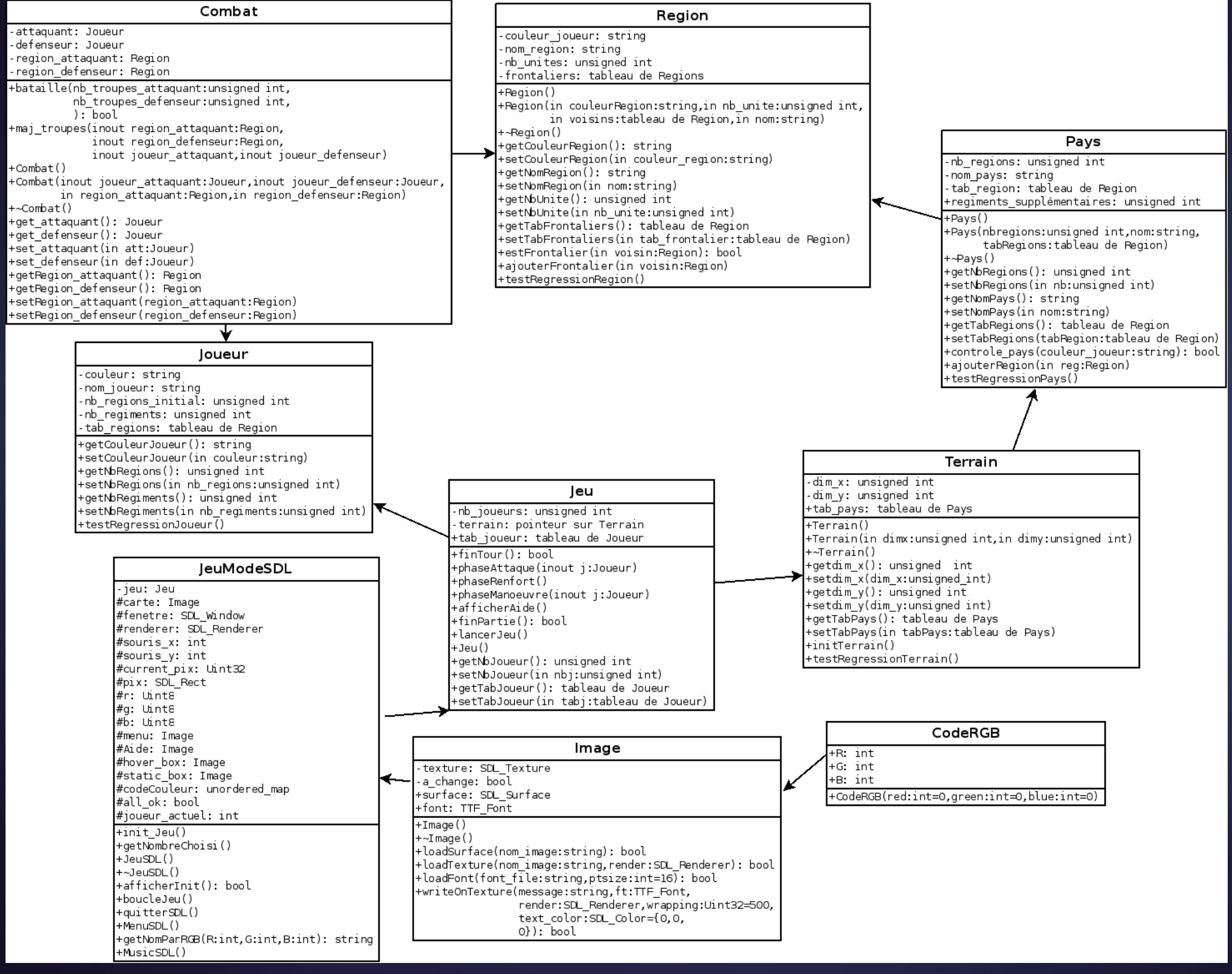
Yannis Hutt
11408376

Julien Cadier
11510421

Randy Andriamaro
11512256

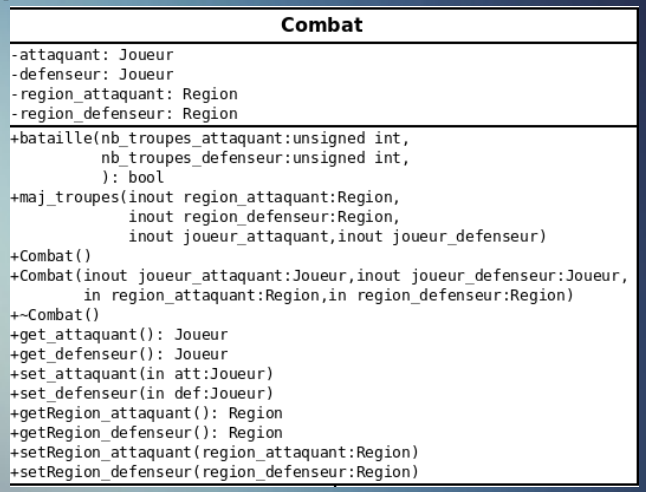
But du jeu

Le jeu est une variante du jeu de plateau Risk. Le jeu est basé sur la stratégie, le but du jeu et de conquérir les régions et les pays, et le but principal du jeu est basée sur la règle de la domination (conquérir toute la carte ou être le joueur le plus puissants ou influant).


Combat
-attaquant: Joueur
-defenseur: Joueur
-region_attaquant: Region
-region_defenseur: Region
+bataille(nb_troupes_attaquant:unsigned int,
nb_troupes_defenseur:unsigned int,
): bool
+maj_troupes(inout region_attaquant:Region,
inout region_defenseur:Region,
inout joueur_attaquant,inout joueur_defenseur)
+Combat()
+Combat(inout joueur_attaquant:Joueur,inout joueur_defenseur:Joueur,
in region_attaquant:Region,in region_defenseur:Region)
+~Combat()
+get_attaquant(): Joueur
+get_defenseur(): Joueur
+set_attaquant(in att:Joueur)
+set_defenseur(in def:Joueur)
+getRegion_attaquant(): Region
+getRegion_defenseur(): Region
+setRegion_attaquant(region_attaquant:Region)
+setRegion_defenseur(region_defenseur:Region)
Region
-couleur_joueur: string
-nom_region: string
-nb_unites: unsigned int
-frontaliers: tableau de Regions
+Region()
+Region(in couleurRegion:string,in nb_unite:unsigned int,
in voisins:tableau de Region,in nom:string)
+~Region()
+getCouleurRegion(): string
+setCouleurRegion(in couleur_region:string)
+getNomRegion(): string
+setNomRegion(in nom:string)
+getNbUnite(): unsigned int
+setNbUnite(in nb_unite:unsigned int)
+getTabFrontaliers(): tableau de Region
+setTabFrontaliers(in tab_frontalier:tableau de Region)
+estFrontalier(in voisin:Region): bool
+ajouterFrontalier(in voisin:Region)
+testRegressionRegion()
Pays
-nb_regions: unsigned int
-nom_pays: string
-tab_region: tableau de Region
+regiments_supplémentaires: unsigned int
+Pays()
+Pays(nbregions:unsigned int,nom:string,
tabRegions:tableau de Region)
+~Pays()
+getNbRegions(): unsigned int
+setNbRegions(in nb:unsigned int)
+getNomPays(): string
+setNomPays(in nom:string)
+getTabRegions(): tableau de Region
+setTabRegions(tabRegion:tableau de Region)
+controle_pays(couleur_joueur:string): bool
+ajouterRegion(in reg:Region)
+testRegressionPays()
Joueur
-couleur: string
-nom_joueur: string
-nb_regions_initial: unsigned int
-nb_regiments: unsigned int
-tab_regions: tableau de Region
+getCouleurJoueur(): string
+setCouleurJoueur(in couleur:string)
+getNbRegions(): unsigned int
+setNbRegions(in nb_regions:unsigned int)
+getNbRegiments(): unsigned int
+setNbRegiments(in nb_regiments:unsigned int)
+testRegressionJoueur()
Jeu
-nb_joueurs: unsigned int
-terrain: pointeur sur Terrain
+tab_joueur: tableau de Joueur
+finTour(): bool
+phaseAttaque(inout j:Joueur)
+phaseRenfort()
+phaseManoeuvre(inout j:Joueur)
+afficherAide()
+finPartie(): bool
+lancerJeu()
+Jeu()
+getNbJoueur(): unsigned int
+setNbJoueur(in nbj:unsigned int)
+getTabJoueur(): tableau de Joueur
+setTabJoueur(in tabj:tableau de Joueur)
Terrain
-dim_x: unsigned int
-dim_y: unsigned int
+tab_pays: tableau de Pays
+Terrain()
+Terrain(in dimx:unsigned int,in dimy:unsigned int)
+~Terrain()
+getdim_x(): unsigned int
+setdim_x(dim_x:unsigned_int)
+getdim_y(): unsigned int
+setdim_y(dim_y:unsigned int)
+getTabPays(): tableau de Pays
+setTabPays(in tabPays:tableau de Pays)
+initTerrain()
+testRegressionTerrain()
JeuModeSDL
-jeu: Jeu
#carte: Image
#fenetre: SDL_Window
#renderer: SDL_Renderer
#souris_x: int
#souris_y: int
#current_pix: Uint32
#pix: SDL_Rect
#r: Uint8
#g: Uint8
#b: Uint8
#menu: Image
#Aide: Image
#hover_box: Image
#static_box: Image
#codeCouleur: unordered_map
#all_ok: bool
#joueur_actuel: int
+init_Jeu()
+getNombreChoisi()
+JeuSDL()
+~JeuSDL()
+afficherInit(): bool
+boucleJeu()
+quitterSDL()
+MenuSDL()
+getNomParRGB(R:int,G:int,B:int): string
+MusicSDL()
Image
-texture: SDL_Texture
-a_change: bool
+surface: SDL_Surface
+font: TTF_Font
+Image()
+~Image()
+loadSurface(nom_image:string): bool
+loadTexture(nom_image:string,render:SDL_Renderer): bool
+loadFont(font_file:string,ptsize:int=16): bool
+writeOnTexture(message:string,ft:TTF_Font,
render:SDL_Renderer,wrapping:Uint32=500,
text_color:SDL_Color={0,0,
0}): bool
CodeRGB
+R: int
+G: int
+B: int
+CodeRGB(red:int=0,green:int=0,blue:int=0)

Classe Combat

- Classe qui permet de gérer la phase de jeu pour les combats, elle permet aux joueurs d'attaquer une région



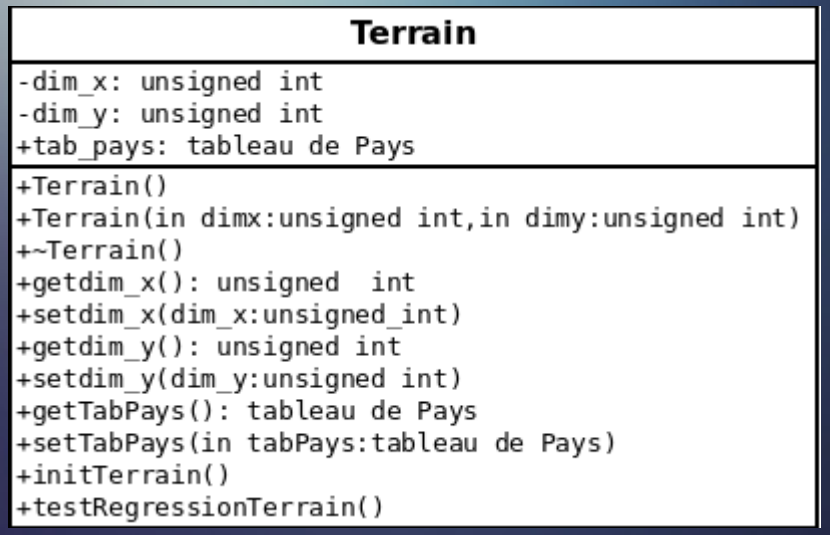
- Constructeur et destructeur
- Accesseur et mutateur

- La fonction bataille prend en arguments le nombre de troupe du joueur qui attaque et du défenseur, et renvoie le vainqueur de la bataille
- La fonction maj_troupe permet de lancer un combat, et permet de faire les modifications suite à la bataille
- Les paramètres de la classe sont un attaquant et un défenseur du type Joueur, ainsi que leurs régions qui sont du type Region

Classe Terrain

- Classe qui permet d’initialiser le terrain, elle a comme paramètre un tableau de pays, ce qui permet après d’ajouter toute les frontières de chaque pays cela permet de faire les tests en suite.

- On trouve les accesseurs et mutateurs
- init_terrain on sont ajouter les frontière.



Classe Jeu

- La classe jeu permet de gérer tous l’algorithme du jeu, ainsi qu' initialiser la partie, avec la répartition aléatoire des pays en fonction du nombre de joueurs, il permet aussi la saisie des noms du joueur, et il lance les trois phases du jeu (phase attaque, phase renfort, phase manœuvre).
- Les fonctions importantes sont phaseAttaque()

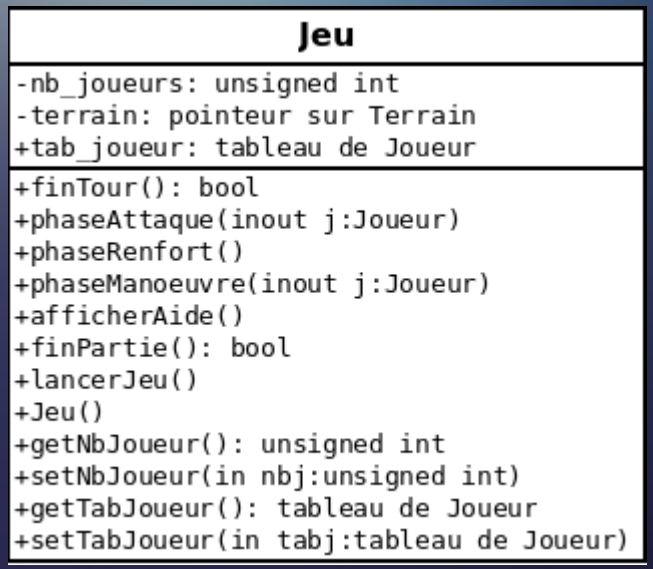
phaseManoeuvre(), et phaseRenfort() qui sont

Les trois phases principales du jeu. Les fonctions

Prennent en argument un joueur pour permettre de

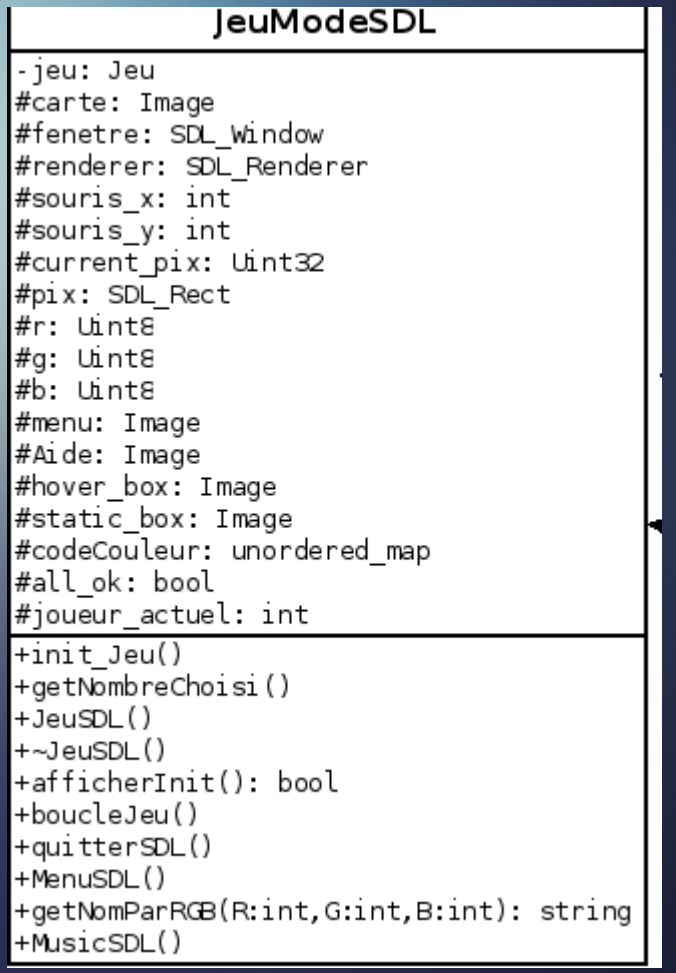
Mettre à jour les données du joueur.

- Accesseur et Mutateur



Classe JeuSDL

- La classe permet de passer du jeu qui était en mode texte, en graphique, on y trouve une fonction qui permet d'initialiser les fonctions de la SDL, ainsi que l'utilisation d'une classe image qui permet de charger les images qui vont être utilisées pour le menu ou le jeu.
- L'identification des régions se déroule grâce au code RGB qui est différent pour chaque pays, quand on clic on récupéré le code a chaque clic pour savoir dans quelle région le joueur a cliquer.



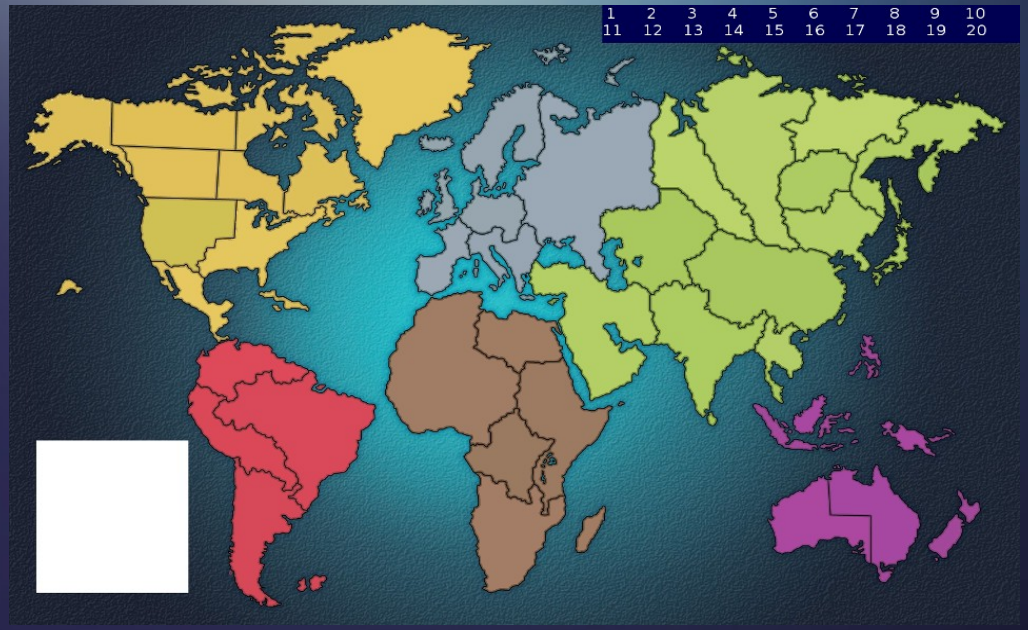
Objectif

- Concevoir le jeu et le faire fonctionner sans bugs dans le temps demandé

(cf diagramme de Gantt)

- Mettre de la musique dans le jeu avec Sdl_mixer

- Disposer d'une interface graphique simple d'utilisation



Complication

- Nous avons eu des complications pour l’utilisation de la musique avec SDL_mixer.
- Passage du mode texte en mode graphique avec la SDL.
- Les problèmes de performance de la SDL 2.0

Piste d’amélioration

- Utilisation d’une autre bibliothèque pour la musique comme FMOD.
- Ajout d’animation sur la carte de jeu.
- Amélioration de l’ergonomie du jeu.
- Amélioration des performances pour l’affichage du jeu.
- Exploration d’autres bibliothèque que SDL, comme la SFML