Graphics est un répertoire contenant toutes les fonctions graphiques gérant l'interface. Nous avons tenu à bien segmenter le code et à respecter les principes du C++ d'arguments privés. Ainsi, aucune fonction ne touche aux structures du jeu. Nous avons seulement deux exceptions, hpallydecrease et hpfoedecrease, qui impactent respectivement la barre de vie du pokémon allié et celle du pokémon ennemi en combat. Ces fonctions modifient la vie du pokémon associé, et aucune autre caractéristique. Le principe d'hermétisme est bien respecté par ailleurs. Lorsqu'un pointeur est envoyé sur un objet d'une classe, ce pointeur est inchangé.

Nous avons distingué deux principales natures de graphiques:

-Field, qui sert à représenter la map du jeu sur laquelle se déplace le personnage. Dans le fichier associé, la génération de la map est gérée, ainsi que le déplacement du personnage et la rencontre de pokémons sauvages. A l'aide de la touche Espace, le joueur peut afficher un menu lui permettant notamment de consulter ses pokémons.

-l'interface de combat, comprenant l'écran de combat et les menus associés. L'interface de combat est créée à l'aide d'une fonction prenant en argument l'arrière-plan (background), la base du pokémon ennemi, celle du pokémon allié, la barre de menus présente en bas, les deux boites de data contenant la vie des pokémon, ainsi que leurs nom, leurs niveau (Level), l'image de leurs barre de vie, et enfin la vie sous forme d'un entier pour le pokémon allié.  
Les menus proposés permettent notamment de choisir le pokémon qui combat, de s'enfuir, ou de choisir ses attaques dans le cas du choix du fight menu. Grâce à un usage de la technique de double buffering, alors même que les menus sont gérés par des fonctions différentes, la transition à l'écran s'effectue sans changement visuel pour l'utilisateur.

La plupart des fonctions utilisées sont des fonctions soit de type void, soit renvoyant un flag. Par exemple, l'affichage d'un menu peut être un flag demandé par un utilisateur, tel le Fight menu lors d'un clic sur Fight. Toutes ces fonctions prennent en paramètre l'écran, et décident tour à tour ce qu'elles appliquent dessus. A l'opposé de ces flags volontaires, certains flags telle la rencontre avec un pokémon, simulée par une loi binomiale, sont involontaires.

Une des grosses difficultés du code était de ne pas avoir une seule fonction massive de plusieurs milliers de lignes. Nous avons donc compartimenté notre code en de nombreuses fonctions, ce qui a également aidé au respect du principe de segmentation du code évoqué plus haut. On peut par exemple l'observer dans la fonction LaunchFoePokemon, où 5 lignes de SET\_BATTLE servent à faire appel à de nombreuses fonctions définies dans Battle.