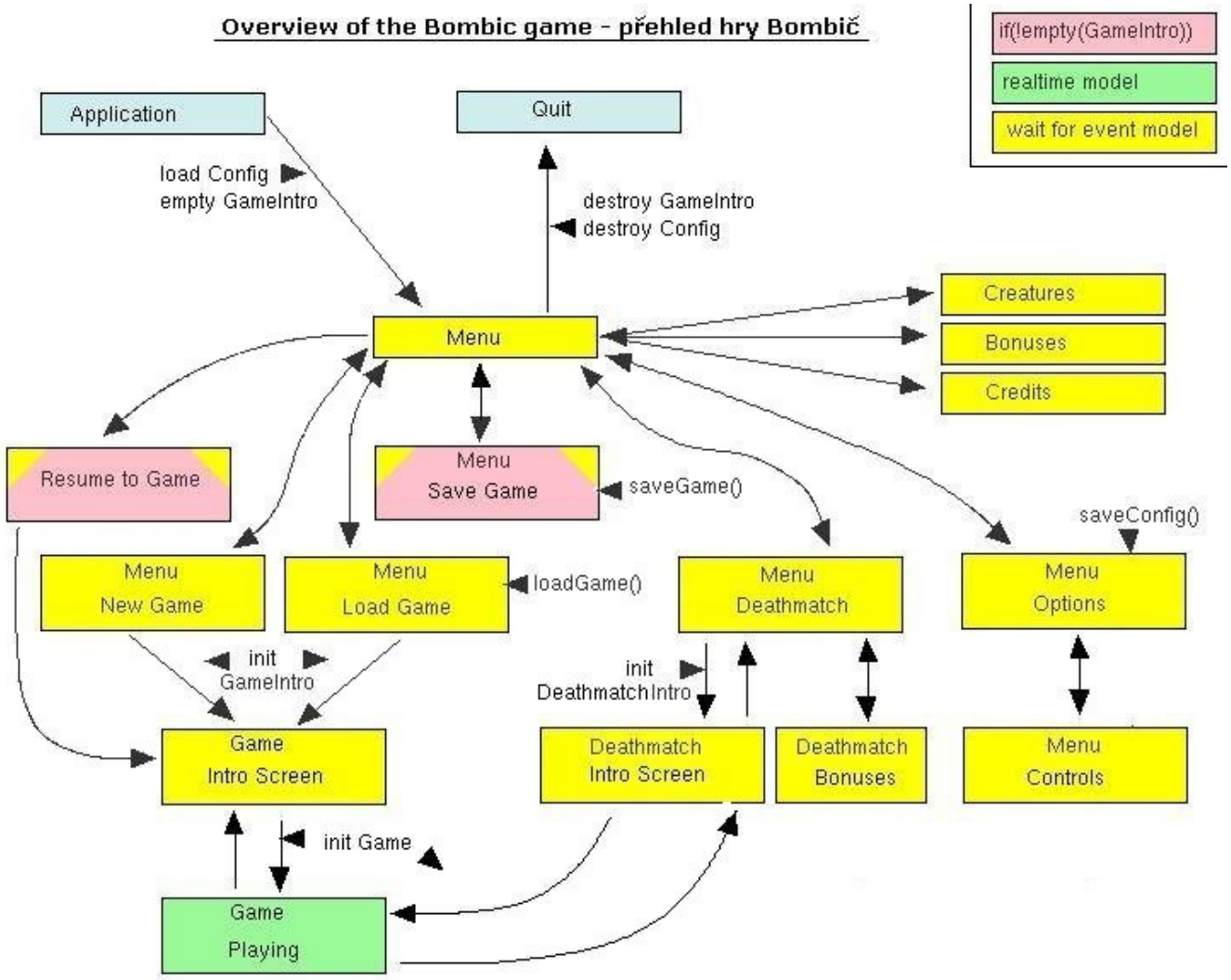
# Návrh rozvržení hry Bombič




Přehled důležitých tříd programu:

*Config*: interface

- load() { načte konfiguraci ze souboru }
- language() { kód zvoleného jazyka }
- visiblepresumtion() { viditelný odhad výbuchu }
- speed() { rychlost hry }
- fullscreen() { celoobrazovkový režim }
- sound() { zapnutý zvuk }
- player(player_num, action) { vrací klávesu nastavenou hráči player_num na akci action }

*GameIntro*: interface

- load(players_count, episode) {vytvoří novou hru}
- load(filename) {načte ze souboru uloženou hru}
- clear() {zničí rozehranou hru}
- level(), inc_level() {vrací respektive navíc zvyšuje dosažený level }
- players() {vrací počet hráčů ve hře }

- player(player_num, &lives, &bombs, &flames, &boots) { proměnné naplní parametry hráče }
- set_player(player_num, lives, bombs, flames, boots) { nastaví parametry hráče }
- mapname() { vrací jméno mapy }

struktury

- většinu dat ukládá do struktury Game

*DeathmatchIntro*: interface

- load(players_count, mapname, bonuses, wins, creatures, bombsatend) {vytvoří novou hru}
- players() { vrací počet hráčů ve hře}
- mapname() { vrací jméno nastavené mapy }

struktury

- většinu dat ukládá do struktury Game

*Game*: interface

- construktor Game(players_count, mapname, deathmatch=false, creatures=true, bombsatend=false) { vytvoří konkretní hru }
- set_player(player_num, lives, bombs, flames, boots) { nastaví parametry hráče }
- play() { spustí hru, fce končí koncem levelu, výhrou deathmatche, nebo prohrou }
- success() { vraci true pokud hra skončila postupem do dalšího levelu }
- player(player_num, &lives, &bombs, &flames, &boots) { proměnné naplní parametry hráče }

struktury

- všechny objekty v mapě budou roztřízeny podle toho jestli jsou v mapě
  - staticky – od začátku do konce mapy, bez pohybu, pouze s možností animace
  - dynamicky – během hry vznikají, zanikají nebo se pohybují, také možnost animace
- mapObject(virtual) { předek všech objektů mapy, čistě virtuální třída }
  - staticMapObject { statický objekt mapy, definici viz výše }
    - type wall { typ zeď, objekt který neshoří, nepohybuje se a nejde přes něj chodit }
    - type floorobject { typ objekt na podlaze, kosmetická drobnost, na hru nemá vliv }
    - type background {prvek pozadí, může se změnit obrázek při výbuchu }
    - method draw() {významná pouze metoda na vykreslení, jinak se měnit nemůžou }
  - dynamicMapObject { dynamický objekt mapy, definici viz výše }
    - type bomber { typ postava bombiče, vytváří bomby, zabíjí, umírá, sbírá bonusy atd. }
    - type creature { typ nestvůra, zabíjí bombiče, umírá, má umělou inteligenci atd. }
    - type box { typ bedna, hoří při zasažení plamenem, nehýbe se, nezabíjí, vytváří bonus }
    - type bomb { typ bomba, vytváří plamen, může se hýbat }
    - type flame { typ plamen, zabíjí, nepohybuje se }
    - method draw() { metoda na vykreslení pouze vykresluje }
    - method move() { metoda na hýbnutí objektem hýbne, také inicializuje řadu dalších přidružených metod jako check() jestli nemá objekt umřít, sebrat bonus, vytvořit jiný objekt atd. }
- list<dynamicMapObject> D { spojový seznam dynamických objektů abych je mohl v konstantním čase vytvářet a hlavně mazat }
- vector<staticMapObject> S {pole statických objektů v podstatě použiji jen abych je měl kam dát}
- vector<vector<list<mapObject*> > > map {dvourozměrné pole jako políčka mapy, současně na jednom políčku může být více objektů, shromáždím je ve spojovém seznamu, opět kvůli rychlému mazání }

předpokládaná implementace

- teoretická implementace metody play():
- while (!is_quit()){
  - pro každý X € D: X.move() { každý X se sám postará o to jestli má umřít, sebrat bonus atd. }
  - pro každý X € map (zleva zhora po řádcích): X.draw() { vykreslí všechny objekty mapy }

    - wait(fps) { zdržovací funkce na přesný počet iterací za sekundu }
- }