class Player:

Syfte med klassen:

Player används för att ta hand om alla metoder för spelaren.

Beskrivning av klassen:

Den används för att kunna röra spelaren samt för att spelaren ska kunna skjuta mot fiender. Den håller koll på hur många liv spelaren har kvar samt sätter en sköld efter en kollision med en fiende.

Relationer:

Player använder arv från Entity och samarbetar med Projectile för skotten.

Konstruktorn:

Player:s konstruktor sätter hur snabbt spelaren åker runt på skärmen och hur många liv spelaren har.
Den anger hur stor kollisionsarea fienderna har att träffa spelaren på.
Den kollar vilka tangenter som är nedtryckta, och om man klickar för att skjuta eller inte.
Sist sätter den spelarens sköld så den är avslagen när spelet börjar.

Publika metoder:

Player har 3 publika metoder som bör förklaras, dessa är:

void move():

Kollar vilka tangenter som är nedtryckna och flyttar spelaren därefter. Kollar även så att spelaren inte åker utanför spelfönstret.

void check_events(SDL_Event&):

Kollar vilka tangenter som trycks ner och sätter dessa till *true*. Om jag t.ex. trycker ned W och A sätts *isPressed["w"]* och *isPressed["a"]* till *true*.
Kollar också om man skjuter med vänster musknapp och om man vill aktivera bomben med höger musknapp.

string get_type():

Returnerar helt sonika vilken typ klassen är, i detta fall kommer "Player" returneras.

Beskrivning av variablar:

map<string, bool>:

Det första argumentet som tar en *string* anger vilken knapp den tillhör
Det andra argumentet anger om denna tangent är nedtryckt eller inte. Anger *true* om den är nedtryckt, annars *false*.

int velocity:

Anger spelarens hastighet på spelplanen. Högre nummer gör så att spelaren rör sig snabbare.

int lives

Anger spelarens liv.

bool shieldUp

Om den är *true* har spelaren kolliderat med en fiende och har en sköld som skyddar spelaren från att ta skada från en fiende under 5 sekunder.

class Nickname

Syfte med klassen:
Sätter spelarens initialer i början av spelet, och även ändrar initialerna om så önskas.

Beskrivning av klassen:
När spelaren startar spelet möts denna av ett fönster som frågar efter dess initialer. Dessa sparas genom spelet tills man stänger av det.

Relationer:
Har en relation med Draw, detta för att kunna använda Draw:s funktion för att skriva ut text på skärmen.

Konstruktorn:
Konstruktorn sätter variablen *nickname* till det som blir inskickat när man skapar en instans av klassen. När man skapar en instans sätts *run* till *true* och det gör att dess loop är aktiv.
Den skickar även med SDL_Surface.

Publika metoder:
Det finns 2 metoder som behöver förklaring:
void HandleEvents(SDL_Event&)
: Kollar vad spelaren vill göra med sina tangenttryckningar.
I *nickname*-klassen kan man använda alla Engelska bokstäver för att skriva sina initialer.
Vi valde att göra så eftersom det känns mer retro om man skippar Å, Ä och Ö.
När spelaren har skrivit in 3st initialer frågar spelet om dessa är OK och om spelaren trycker på *Y* kommer spelaren in i menyn, om inte kan spelaren trycka på Backspace och sudda ut en initial åt gången.

void RunNickname()
: Kör loopen där man kan skriva in sina initialer. Häri skrivs initialerna ut på skärmen m h a DrawText från Draw-klassen.

Beskrivning av variablar:
SDL_Surface displaySurface*
: Används för att kunna använda den befintliga skärmen som allt ritas ut på.

SDL_Event Events
: Kollar när använder försöker interaktera med spelet

string nickname
: Håller spelarens initalier

bool run
: Om denna är *true* körs loopen

Diskussion över vårt upplägg:

Vi hade ingen riktig plan hur vi tänkte lägga upp designen utan körde mer på hur det blev i slutändan, men såhär i slutet blev den inte riktigt optimal tycker vi. Vi har pratat med några andra grupper och åtminstone en annan grupp har bara en loop där dom kör sina olika klasser, inte som vi har med loopar i nästan varje klass.
Nackdel med vår design är att den är väldigt svåröverskådlig eftersom vi har en SDL_Surface som måste skickas med överallt, samt en SDL_Delay() och en SDL_Flip() i varje loop, alltså i varje klass. Om man hade gjort som den andra gruppen med bara en loop hade man fått en mycket bättre överblick samt mindre kod att hålla reda på.
Om man ska säga något positivt är väl att varje del av vår kod *kan* fungera själv om man verkligen vill, dock finns det ingen anledning till detta.
Vår anledning till att vi “valde” att ha vår design som den är var att båda var rätt färska på OO-programmering och visste väl inte riktigt hur vi skulle tänka för att få allting så optimalt som möjligt, men den som lever den lär.

Externa filformat:

De enda externa filformat som vi använt oss utav är .bmp som vi använder till vår spelare, våra fiender, skotten och hur liven visas.