# Bestudeer de video van de volgende les geheel:

## Templates

Experimenteer zelfstandig met de daarin behandelde voorbeeld-programma's. Een link naar de broncode daarvan staat onder de video. In de volgende les kun je hierover vragen stellen.

# Opgaven bij de huidige les:

In te leveren uiterlijk 2 dagen vóór de volgende les.

**Het programma heeft een voorgeschreven opbouw.** Verbeteringen hierop zijn toegestaan in overleg met de docent. Begin je bron-bestand weer met requirements, testspecs en design.

Dit programma bouwt voort op het huiswerk van de vorige les. Breidt het programma uit zodat het nu de volgende klassen bevat:

***class Vec3D**: O*ngewijzigd t.o.v. vorige programma.

***class Object***: Dit is een abstract base class met een field *Vec3D center* en een constructor met signature:

```
Object (float x, float y, float z): center (x, y, z) {}
```

Daarnaast heeft deze class een pure virtual method met signature:

```
bool hit (Ray &ray)
```

Deze method test in derived classes of het betreffende object geraakt wordt door de straal en wat de *support* vector en de *direction* vector van de weerkaatste straal zijn.

***class Ray***: Deze class heeft naast de fields van *class Ray* uit de vorige opgave field *VPO &objects* waarbij *VPO* gedefinieerd is als *st::vector <Object*>*. Daarnaast heeft deze class methods met de volgende signatures:

```
Ray (float xStart, float yStart, VPO &objects)
```

en

```
bool scan ()
```

De *scan* method roept voor alle objecten met parent class *Object* de *hit* functie aan. Indien een *Ray* een *Object* raakt, wordt deze *Ray* niet verder gevolgd.

***class Sphere***: Deze class erft public van class *Object* en implementeert de *hit* functie op een manier die past bij een bol. Daartoe heeft hij onder andere de *hitPoint* method die net zo werkt als in de vorige opgave. De *hit* functie kan echter meer, zoals beschreven bij de parent class.

***class Floor***: Ook deze class erft public van class *Object* en implementeert de *hit* functie op een manier die past bij een plat vlak met een schaakbord-patroon. De lichtgekleurde velden weerkaatsen de *Ray.* De donker gekleurde velden laten hem ongehinderd door en gedragen zich dus eigenlijk als gaten in het schaakbord, d.w.z. *hit* levert *false* indien de *Ray* op een donker gekleurd veld valt.

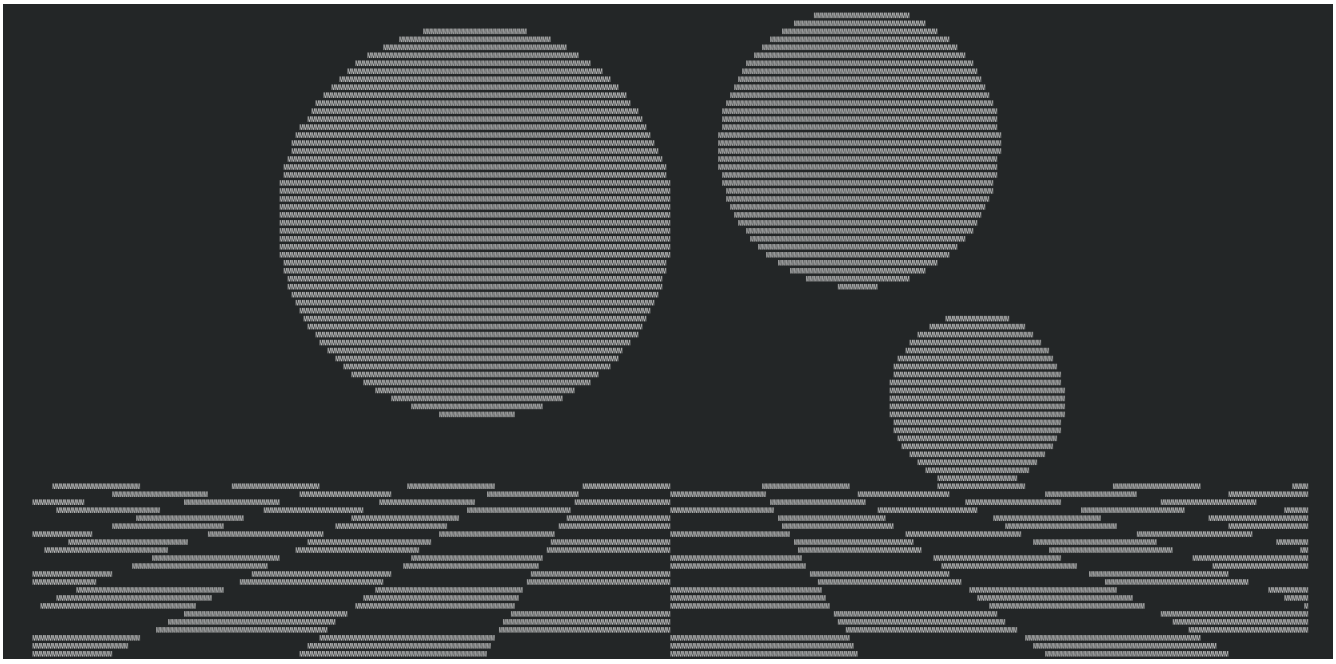
***class RayScanner***: De enige instance van deze class bevatten een lijst van 4 *Object*'s, namelijk 1 *Floor* en 3 *Sphere*'s van verschillende afmetingen en op verschillende posities.

Daarnaast bevat de *RayScanner* een method met signature:

```
void scan ()
```

Deze lanceert vanaf een gezichtspunt 3 meter (SI eenheden by default) achter een denkbeeldig beeldscherm *Ray*'s, eentje door elke pixel van dit scherm dat een breedte heeft van ca 80 pixels en een hoogte van ca 40 pixels.

Vervolgens worden op dezelfde wijze als bij het huiswerk van les 1 hieruit een beeld op de console gegenereerd dat er uitziet zoals onderstaand. Zoom in om te kijken hoe het is opgebouwd.



Deze opgave is het laatste station voor de eindopgave! Referenties naar de wiskunde die nodig is voor de berekening van *support* en *direction* van de weerkaatste straal staan in het document *les_inhoud.pdf* van deze les.

De beoordeling van deze huiswerkopgave maakt deel uit van je eindcijfer. Net als in bij de vorige opgaven wordt niet alleen beoordeeld of je programma wel of niet werkt, maar vooral of je zowel in grote lijnen als in details begrepen hebt wat je aan het doen bent, inclusief de wiskunde, en de gemaakte keuzen kunt motiveren.

Programma en verstandig gedoseerd commentaar weer in het engels. Zet verwijzingen naar gebruikte on-line informatie-bronnen weer als commentaar in je programma.

--//--