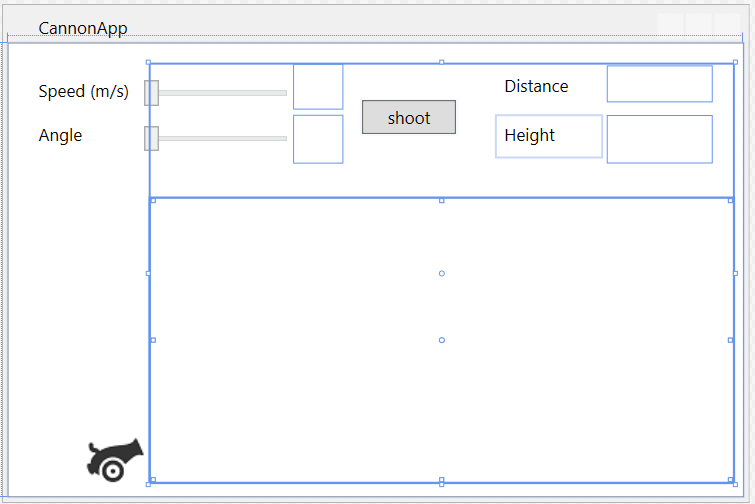
# Setup

Dus de opgave houdt in dat we een cannonball application gaan maken. Het gaat over één window zoals op de foto hieronder, die twee sliders, twee labels (die de waarde van de respectievelijke slider bevat). Vervolgens is er een button waarmee de kanonbal zal worden afgeschoten. Dan zijn er ten slotte ook nog 4 labels, een met “height” en “distance” in en dan twee corresponderende labels met de waarde van de hoogte en de afstand van de kanonbal. Onder die bovenzijde is dan nog een foto van het kanon en een canvas (width: 410, height: 200).

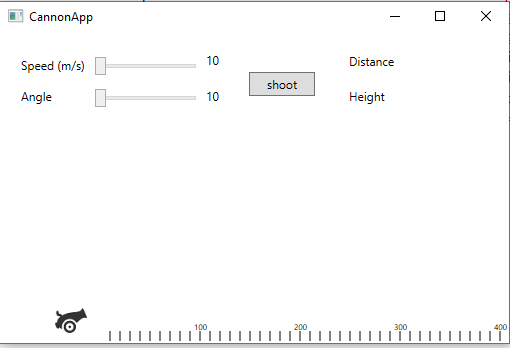


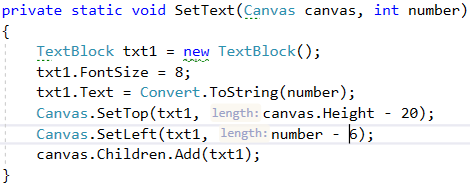
## ZORG EERST DAT JE DEZE LAYOUT VOLLEDIG IDENTIEK HEBT

Zorg daarbij ook dat de waardes van de sliders ook al in de labels worden getoond.

# De klasse World

Als alles op het scherm aanwezig is dan zal eerst het volgende scherm moeten worden bekomen, dit gaan we stap voor stap overlopen in onderstaande puntjes.

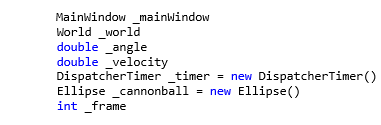


1. Maak een nieuwe klasse in je project aan en noem deze klasse World
   1. Deze klasse zal instaan voor het verzorgen van het uiterlijk van het canvas (de lengte streepjes onderaan met de getallen erbij), ook zal deze klasse een punt transformeren van assenstelsel
   2. Het punt (0,0) op een canvas ligt in de linkerbovenhoek van een canvas, voor onze kanonbal gaan we echter willen dat het punt (0,0) zich bevindt in de linkeronderhoek zoals bij een normaal assenstelsel. We gaan later ook berekeningen doen alsof we werken met een normaal assenstelsel om dit punt daarna om te rekenen naar de locatie dat dit op het canvas inhoudt
2. We gaan beginnen met de klasse drie private variabelen te maken eentje voor de hoogte van het canvas, een voor de breedte van het canvas en ten slotte nog een voor het canvas zelf.
3. Vervolgens gaan we de constructor maken, in deze constructor gaan de private variabelen eerst hun waarde worden toegekend voor er verder iets gebeurt.
4. Daarna zullen eerst alle lijntjes worden getrokken, deze zijn 10 pixels hoog en staan om de 10 pixels geplaatst beginnend bij coördinaat (10,0) in ons “normaal” assenstelsel
   1. Voor het trekken van een lijn refereer ik door naar de methode DrawLine op pagina 87 in het boek
   2. Zorg ook dat de lijntjes altijd zullen worden getrokken over de volledige breedte van het canvas. DIT MAG DUS NIET INGESTELD STAAN OP 400 OMDAT DIT IN HET VOORBEELD ZO IS.
5. Vervolgens gaan we dan de getallen ook plaatsen op het assenstel, dit gebeurt duidelijk om de 100 pixels (of 10 lijntjes)
   1. Gebruik volgende methode om het jezelf makkelijk te maken
6. Als laatste ding in deze klasse maken we dan nog een publieke methode genaamd ConvertPoint, die een Point object meekrijgt. Het doel van deze methode is om een punt dat in het assenstelsel met de linkeronderhoek als oorsprong te transformeren naar een punt op het canvas
   1. Denk dus na waar het punt (0,0) ligt op het canvas
   2. Probeer dit een aantal keer door iets te plaatsen op het canvas als dit niet lukt
7. Dan is onze World klasse af het Window zou dan door het maken van een World object na de oproeping van de methode InitializeComponent(); er uit moeten zien als de foto op pagina 2.

# De klasse Cannonball

We gaan dan nu de klasse maken die volledig de controle zal nemen van het maken en ondersteunen van de kanonbal op het canvas. We maken dus de klasse Cannonball in het project.

1. Deze klasse zal verschillende private variabelen hebben, namelijk de volgende:



1. De variabelen MainWindow en World worden meegegeven in de constructor van de klasse en de private variabelen krijgen daar hun waarde.
2. Vervolgens heeft de klasse 1 publieke methode genaamd Shoot. Deze methode zal vooreerst zorgen dat de sliders en de button in de mainwindow zal gedisabled worden
   1. Voor dit te doen denk aan de private variabele mainWindow voor dit te doen;
3. Vervolgens gaan het onze kanonkogel maken en al zetten op het canvas, het startpunt van onze kanonkogel is op breedte 0 en hoogte 8 (dit is in het assenstelsel met de oorsprong in de linkeronderhoek als je de methode ConvertPoint correct hebt gemaakt, kan je deze nu gebruiken voor niet meer de berekening te moeten maken iedere keer als je moet omvormen voor je kogel op het canvas te zetten of te verplaatsen)
   1. De kanonkogel heeft een breedte en hoogte van 10
4. Als dit allemaal gedaan hebben gaan we gebruik maken van de klasse DispatcherTimer, een deel van het vak .Net is dat je ook jezelf kan inlezen op een nieuwe klasse, onderaan pagina 5 staat een stappenplan over deze klasse als je de volgende link verwarrend vindt. Zorg ook dat je kan werken met je hulp in visual studio zelf (druk eens f1).
   1. <https://docs.microsoft.com/en-us/dotnet/api/system.windows.threading.dispatchertimer?view=netframework-4.8>
   2. Bekijk in het specifiek zeker de property Interval, de methodes Start en Stop en het Tick event
   3. Zorg dat het Tick event wordt opgeroepen iedere 20 milliseconden
   4. Vooraleer je de timer start zorg dat de variabele frame gelijk is aan 0
5. De methode die aan het Tick event gekoppeld is zal UpdateCannonball heten. Hieronder staat het stappenplan voor het maken van deze methode.
   1. Eerst gaan we de variabele frame ophogen met 1 (deze variabele zal aanduiden hoeveel tijd er voorbij is gedaan in 1/50ste secondes)
   2. Vervolgens maken we de nieuwe locatie van de kanonkogel dit doen we aan de hand van de volgende formules:
      1. X = velocity \* Cosinus(angle (in radialen)) \* (frame / 50.0)
      2. Y = velocity \* Sinus(angle) \* (\_frame / 50.0) - 0.5 \* 9.81 \* (\_frame / 50.0)² + 8
6. Voor we de kanonkogel gaan aanpassen, gaan we eerst kijken of deze nog op het canvas zal zijn, als de hoogte van kanonkogel kleiner is dan 0 of als de afstand groter is dan de breedte van het canvas dan zal de timer worden gestopt, zal de kogel verdwijnen van het canvas en zullen de sliders en de button terug actief worden
7. Als dit niet het geval is dan zal de locatie van de kogel worden vertaald naar de locatie op het canvas en zal deze nieuwe locatie gekoppeld worden aan de kanonkogel
   1. Hiervoor moet slechts 1 property van de kanonkogel worden aangepast, er moet geen nieuwe ellips worden gemaakt

Als dit alles is gedaan dan moet enkel nog voor het clickevent van de button een methode worden gemaakt en moet in deze methode een Cannonball object worden gemaakt en moet de Shoot methode worden opgeroepen. Dan zou de volledige applicatie af moeten zijn.

# Extra hulp (consulteer enkel als je zelf het antwoord niet kan vinden)

## De klasse DispatcherTimer

Maak volgende tekst terug groot, deze staat nu op lettertype 1.

De klasse DispatcherTimer is bedoeld voor het coördineren van getimede zaken, bijvoorbeeld als iedere seconde een object van kleur zou moeten verwisselen.

### De property Interval

De property interval gaat zeggen hoe lang het duurt voor het event Tick moet worden gefired dit kan je instellen op vrijwel alles, een minuut, een seconden, een deel van een seconden…

Het instellen van de property Interval doe je door de Klasse TimeSpan aan te roepen door bijvoorbeeld te zeggen

DispatcherTimer timer = new Dispatchertimer();

timer.Interval = TimeSpan.FromSeconds(1);

Dit zou de timer op 1 seconde zetten.

### De methode Start

Redelijk vanzelfsprekend met deze methode start je de timer pas vanaf dan zal na ieder verloop van het interval het Tick event worden gefired tot de methode Stop wordt aangeroepen.

### De methode Stop

De methode Stop zorgt dat de timer stopt met lopen, daarna kan met start de timer terug worden opgestart als dit wordt gevraagd.

### Het Tick event

Het Tick event zorgt ervoor dat er een methode kan worden gekoppeld aan het verlopen van de timer dit kan bijvoorbeeld op de volgende manier.

DispatcherTimer timer = new Dispatchertimer();

timer.Tick += UpdateCannonball;

## De calculatie van de nieuwe locatie van de kanonkogel

Voor het bereken van de methode gaan we verschillende keren beroep doen op de klasse Math hieronder staan alle entries verder uitgelegd op lettertype 1.

### Het omzetten van de hoek

Voor het omzetten van de hoek van graden naar radialen gaan we de hoek vermenigvuldigen met Pi en delen door 180. Dit doen we op de volgende wijze.

Math.PI \* \_angle / 180.0

### Het berekenen van de cosinus/sinus

De cosinus en sinus zijn gewoon methodes in de Math klasse zelf, maar hebben dus de hoek in radialen nodig, het berekenen gebeurt op de volgende wijze

Math.Cos(angle)

Math.Sin(angle)

### De macht laten berekenen

Je kan natuurlijk voor een kwadraat gewoon x \* x doen, maar voor hogere machten is dit niet interessant natuurlijk in de klasse Math zit de methode Pow voor Power, eerst wordt daarin het grondtal gezet en daarna de exponent dus voor x² te verkrijgen geven we dan in

Math.Pow(x,2)