Uitwerkingen eerste deeltentamen Gameprogrammeren
Vrijdag 26 september 2014, 8.30-10.30 uur

---

1. Deze opgave bestaat uit een aantal vragen. Houd het antwoord kort: één of twee zinnen per onderdeel kan al genoeg zijn.

   (a) *(2 punten)* Wat is een statische membervariabele? Geef een voorbeeld in de context van games waarin een statische membervariabele nuttig is.

   Antwoord: Een statische membervariabele is een variabele (een stukje geheugen met een naam) die niet bij een object hoort, maar bij een klasse. Zo'n variabele kan nuttig zijn om een object voor te stellen waarvan er maar eentje is maar die wel overal toegankelijk moet zijn, zoals het object dat de gamewereld voorstelt, of een willekeurige getallengenerator.

   (b) *(3 punten)* Kruis aan welke van de volgende stellingen waar zijn:

   - ☐ Elke klasse moet een statische methode hebben die de naam `Main` draagt. *Nee, precies één klasse moet een statische methode hebben met die naam.*
   - X Een object is een groepje variabelen.
   - X Objecten met een klasse als type worden als verwijzing doorgegeven, objecten met een struct als type worden als waarde doorgegeven.
   - X Een methode met een lege body heeft altijd als resultaatwaarde **void**.
   - ☐ Een opdracht is een stukje programmacode met een waarde. *Nee, een expressie is een stukje programmacode met een waarde.*
   - ☐ Elke imperatieve taal is ook procedureel. *Nee. Een voorbeeld van een imperatieve, niet-procedurele taalgroep is de groep Assembler-talen.*

   (c) *(2 punten)* Als we de waarde van een integer variabele `i` willen toekennen aan een double variabele `d`, dan doen we dat met de toekenning `d = i`. Andersom (toekennen van double waarde `d` aan `i`) ziet er iets anders uit: `i = (int)d;`. In beide gevallen vindt een typeconversie plaatst. Wat is een ander woord voor zo'n typeconversie? En waarom moet er in het tweede geval **(int)** voor de rechterzijde van de toekenning geschreven worden?

   Antwoord: Een typeconversie wordt ook wel een *cast* genoemd. Als een **double** naar een **int** geconverteerd wordt, dan worden de decimalen achter de komma weggegooid (bijvoorbeeld 12.3 wordt 12). Met de cast **(int)** geeft de programmeur expliciet aan dat dit 'akkoord' is. Andersom is dit niet nodig. Een **int** past altijd in een **double**, dus er wordt geen informatie weggegooid.

   (d) *(3 punten)* Hieronder staan zes programma-fragmenten. Je mag er vanuit gaan dat alle variabelen die gebruikt worden vantevoren gedeclareerd en van het type **int** zijn. Geef in elk van de gevallen aan *hoe vaak* de methode `iets` wordt aangeroepen. Licht het antwoord kort toe.

1	**for** (x=1; x<=6; x+=2)
	  this.iets();
	for (y=0; y>5; y++)
	  this.iets();
		3 keer: voor x is 1, 3 en 5, en daarna 0 keer, want de tweede loop stopt onmiddellijk (y is nooit groter dan 5).

2	**for** (x=0; x<5; x++) **for** (y=1; y<=x; y++) **this**.iets();	10 keer: 0+1+2+3+4 keer.

3	**for** (x=0; x<10; x++)
	  if (x % 2 == 0)
	  {
	  this.iets();
	  }
		5 keer: voor alle even waarden van x, dus 0, 2, 4, 6, 8.

4	**while** (**false**);
	  if (**true**)
	  this.iets();
		1 keer: de (lege) **while** body wordt niet uitgevoerd, daarna wordt de body van de **if** 1 keer uitgevoerd.

5	**for** (x=1; x<=5; x+=2)
	  for (y=0; y<5; y+=x)
	  this.iets();
		8 keer: 5+2+1 keer.

6	**for** (x=1; x<18; x+=x)
	  for (y=0; y<x; y++)
	  this.iets();
		31 keer: 1+2+4+8+16.

2. *($\frac{1}{2}$ punt per goed antwoord)* Hieronder staat 20 fragmenten uit een programma. Schrijf in de tabel hieronder achter elk programmafragment één daarbij passende letter, als volgt:

   - **T** als het programmafragment een **type** is
   - **E** als het programmafragment een **expressie** (maar geen constante) is
   - **O** als het programmafragment een **opdracht** is
   - **D** als het programmafragment een **declaratie** is
   - **C** als het programmafragment een **constante** (en dus ook een expressie) is
   - **X** als het programmafragment geen van bovenstaande dingen is

Antwoord:

int i, j;	D	position.Zero;	X	Color.White	E	**true**=**false**	X
(**float**)	X	'\\'	C	Game	T	**for**(;**false**;);	O
Vector2 position;	D	**new** Random()	E	1 + 2 == 3	E	i = Color.Black.R;	O
new Vector2(1, 2) * 3	E	i<5!=j>5	E	SpriteBatch	T	1E1	C
while(**true**) **while**(**false**);	O	/*hallo	X	(**bool**)**true**	E	i = j++;	O

3. (a) *(5 punten)* Schrijf een methode cijfer met twee getallen als parameter. Als de eerste parameter 0 is, geeft de methode het *laatste cijfer* van de tweede parameter terug, als de eerste parameter 1 is, geeft de methode het *voorlaatste cijfer* van de tweede parameter terug, als de eerste parameter 2 is, geeft de methode het *op twee na laatste cijfer* van de tweede parameter terug, enzovoorts.

   Bijvoorbeeld: cijfer(0,456) geeft 6 terug, cijfer(2,98765) geeft 7 terug, en cijfer(4,1234) geeft 0 terug. Je mag zonder controle aannemen dat beide parameters niet negatief zijn.

   Antwoord:

```
1  public int cijfer(int n, int x)
2  {
3      int t = 0;
4      while (t < n)
5      {
6          x = x / 10;
7          t++;
8      }
9      return x % 10;
10 }
```

   (b) *(5 punten)* Bankrekeningnummers (met uitzondering van de vroegere gironummers) bestaan uit 9 cijfers. Maar niet elk getal van 9 cijfers is een geldig rekeningnummer. Om te controleren of er geen tikfouten zijn gemaakt bij het invoeren van rekeningnummers, doet online banking software de volgende controle: Het eerste cijfer wordt vermenigvuldigd met 9, het tweede cijfer wordt vermenigvuldigd met 8, het derde cijfer met 7, enzovoort, en het laatste cijfer met 1. Alle uitkomsten worden opgeteld. Als het totaal deelbaar is door 11, is het een geldig rekeningnummer. Bijvoorbeeld: voor de controle van 839801149 wordt uitgerekend: $8 \times 9 + 3 \times 8 + 9 \times 7 + 8 \times 6 + 0 \times 5 + 1 \times 4 + 1 \times 3 + 4 \times 2 + 9 \times 1 = 231$, en dat is inderdaad deelbaar door 11.

   Schrijf een methode controleer met een getal als parameter, dat teruggeeft of dat getal een geldig bankrekeningnummer is. Vermijd daarbij om 9 maal vrijwel dezelfde expressie op te schrijven; gebruik in plaats daarvan een C#-opdracht om de regelmaat uit te buiten. Je mag hierbij de methode cijfer uit de vorige deelvraag hergebruiken.

   Antwoord:

```
1  public bool controleer(int x)
2  {
3      int t = 0;
4      int s = 0;
5      while (t < 9)
6      {
7          s = s + (t+1)*cijfer(t, x);
8          t++;
9      }
10     return s % 11 == 0;
11 }
```

4. (a) *(1 punt)* Wat is het type van het object waar **this** naar wijst in regel 15 in de klasse AsteroidShooter?

Antwoord: AsteroidShooter

(b) *(1 punt)* De ufo beweegt van boven naar beneden mee met de muispointer, maar blijft altijd aan de linkerkant van het scherm (zie ook de screenshot), en beweegt nooit uit het scherm. Schrijf de body van de HandleInput methode van de Ufo klasse op.

Antwoord:

```
1  position.Y = MathHelper.Clamp(inputHelper.MousePosition.Y, 0, AsteroidShooter.Screen.Y − sprite.Height);
```

(c) *(2 punten)* De Missile klasse heeft ook een HandleInput methode. Als de speler op de linkermuisknop klikt, dan moet de raket een positieve x-snelheid krijgen van 400, maar dat mag alleen als de raket niet al aan het vliegen is. Werk de body van deze methode uit.

Antwoord:

```
1  if (inputHelper.MouseLeftButtonPressed() && velocity == Vector2.Zero)
2      velocity = new Vector2(400, 0);
```

(d) *(3 punten)* De Update methode van Missile moet ervoor zorgen dat de positie van de raket wordt bijgewerkt. Werk de body van deze methode uit. Zorg dat alle mogelijke gevallen afgehandeld worden, zoals wat er gebeurt als de raket uit het scherm vliegt. Als de raket nog niet is afgeschoten, dan moet hij achter de ufo getekend worden, zodat hij niet zichtbaar is. Let op: we handelen in deze methode geen botsingen met asteroïden af.

Antwoord:

```
1  if (velocity == Vector2.Zero)
2      position = AsteroidShooter.GameWorld.Ufo.Position + new Vector2(10, 20);
3  else
4      position += velocity ∗ (float)gameTime.ElapsedGameTime.TotalSeconds;
5  if (position.X > AsteroidShooter.Screen.X)
6      velocity = Vector2.Zero;
```

(e) *(3 punten)* In de Update methode van Asteroid moet een aantal dingen gebeuren. Ten eerste moet de positie van de asteroïde aan de hand van de snelheid worden aangepast. Als de asteroïde links uit het scherm gevlogen is, dan moet hij een nieuwe (willekeurige) positie krijgen rechts buiten het scherm. Om ervoor te zorgen dat asteroïden niet altijd op hetzelfde moment het scherm in en uit vliegen moet deze nieuwe positie niet altijd onmiddellijk toegekend worden, maar na een willekeurig aantal iteraties van de game loop. Als de asteroïde botst met een raket, dan moeten de asteroïde en de raket ge-reset worden. Werk de body van deze methode uit.

Antwoord:

```
1  position += velocity ∗ (float)gameTime.ElapsedGameTime.TotalSeconds;
2  if ((position.X < −sprite.Height && AsteroidShooter.Random.NextDouble() < 0.01))
3  {
4      double yPos = AsteroidShooter.Random.NextDouble() ∗ (AsteroidShooter.Screen.Y − sprite.Height);
5      position = new Vector2(AsteroidShooter.Screen.X, (float)yPos);
6  }
7  if (this.CollidesWith(AsteroidShooter.GameWorld.Missile))
8  {
9      Reset();
10     AsteroidShooter.GameWorld.Missile.Reset();
11 }
```