Uitwerkingen derde deeltentamen Gameprogrammeren
Vrijdag 9 november 2012, 8.30-10.30 uur

---

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

   (a) *(3 punten)* De volgende klasse is onderdeel van de library GameManagement:

   ```
   public class InputHelper
   {
       // ...
   }
   ```

   Wat betekent het woord **public** voor de klasse? Wat is het gevolg als we dat woord weg zouden laten?

   Antwoord: het woord **public** betekent dat de klasse toegankelijk is buiten zijn eigen assembly. Met andere woorden: andere applicaties kunnen deze klasse ook gebruiken. Als we het woord **public** weg zouden laten, dan is de klasse standaard *intern*, oftewel: hij kan alleen in zijn eigen assembly (GameManagement) gebruikt worden.

   (b) *(2 punten)* Eén van de volgende drie declaraties-met-toekenningen is correct. Welke is dat, en waarom zijn de andere twee niet correct?

   ```
   List<int> a = new IList<int>(); // versie 1
   IList<int> b = new List<int>(); // versie 2
   IList<int> c = new IList<int>(); // versie 3
   ```

   Antwoord: In versie 1 en 3 wordt een object van IList gecreëerd, maar dat kan niet want IList is een interface en geen klasse. Versie 2 is correct, want klasse List implementeert de interface IList, en mag dus die rol vervullen.

   (c) *(3 punten)* Beschouw de volgende twee klassen:

   ```
   class A {
       public void func1() { Console.WriteLine("A::func1"); }
       public virtual void func2() { Console.WriteLine("A::func2"); }
   }
   class B : A {
       public void func1() { Console.WriteLine("B::func1"); }
       public override void func2() { Console.WriteLine("B::func2"); }
   }
   ```

   Wat is de uitvoer van de volgende serie opdrachten?

   ```
   A x = new A();
   A y = new B();
   B z = new B();
   x.func1();
   x.func2();
   y.func1();
   y.func2();
   z.func1();
   z.func2();
   ```

   Antwoord:

   ```
   A::func1
   A::func2
   A::func1
   B::func2
   B::func1
   B::func2
   ```

(d) *(3 punten)* Beschouw de volgende klassen:

```
abstract class GameObject {
    protected Vector2 position;
    protected Vector2 velocity;
}
class SpriteGameObject : GameObject {
    protected Texture2D sprite;
}
```

Kruis aan welke van de volgende instructies uitgevoerd mogen worden in een methode van een andere, ongerelateerde klasse:

- X `GameObject obj;`
- ☐ `GameObject obj = new GameObject();`
- X `GameObject obj2 = new SpriteGameObject();`
- X `GameObject obj3 = new SpriteGameObject() as GameObject;`
- X `GameObject[] lijst;`
- X `GameObject[] lijst = new GameObject[10];`

(e) *(2 punten)* Wat is een *partial class*? Wanneer is het nuttig om een partial class te gebruiken?

Antwoord: Een partial class is een klasse waarvan de definitie verspreid is over meerdere files. Dit is nuttig om te gebruiken als een klasse heel groot is, maar het niet logisch is om de code te splitsen in verschillende klassen.

(f) *(2 punten)* Kruis aan welke van de volgende stellingen waar zijn:

- ☐ Net zoals bij interfaces mag je ook van meerdere abstracte klassen erven.
- X Nadat een object van het type **string** gemaakt is, mag dat object niet meer veranderd worden.
- X De alpha-testfase is over het algemeen intern (door developers gedaan), de beta-testfase is extern (grote groep spelers).
- ☐ Excepties gebruik je vooral om het programma te stoppen in het geval dat in de code een bug zit die geintroduceerd is door de programmeur.

2. (a) *(4 punten)* Werk de header en body van de readLevel methode uit. Gebruik de **switch** opdracht om de verschillende soorten objecten af te handelen.

Antwoord:

```
public void readLevel(string path)
{
    StreamReader fileLezer = new StreamReader(path);
    string regel = fileLezer.ReadLine();
    while (regel != null)
    {
        char[] splitArr = { ' ' };
        string[] obj = regel.Split(splitArr);
        Vector2 position = new Vector2(float.Parse(obj[1]), float.Parse(obj[2]));
        switch(obj[0])
        {
            case "asteroid": gameObjects.Add(new Asteroid(asteroid, position)); break;
            case "planet1": gameObjects.Add(new SpriteGameObject(planet1, position)); break;
            case "planet2": gameObjects.Add(new SpriteGameObject(planet2, position)); break;
            default: throw new IOException("Unknown object type: " + obj[0]);
        }
        regel = fileLezer.ReadLine();
    }
    fileLezer.Close();
}
```

(b) *(2 punten)* Indien de readLevel methode een exceptie werpt, dan crasht de game. Hoe moeten we de GameWorld constructor aanpassen zodat dit niet meer gebeurt?

Antwoord: we moeten de aanroep van de readLevel methode in een **try**-blok plaatsen en met het **catch**-gedeelte de exceptie afvangen, bijvoorbeeld:

```
try
{
    readLevels();
}
catch(IOException e)
{
}
```

(c) *(4 punten)* Werk de Update methode van Asteroid verder uit, zodat de positie wordt aangepast en wrapping plaatsvindt. Let hierbij op dat de wrapping netjes geïmplementeerd is, oftewel: geen asteroiden die al verplaatst worden als ze nog niet helemaal het scherm uitgevlogen zijn, of asteroiden die al deels in het scherm staan als ze weer tevoorschijn komen.

Antwoord:

```
public override void Update(GameTime gameTime)
{
    this.velocity *= 1.00001f;
    base.Update(gameTime);
    if (this.position.X + this.sprite.Width < 0)
        this.position.X = Asteroids.Screen.X;
    else if (this.position.X > Asteroids.Screen.X)
        this.position.X = -this.sprite.Width;
    if (this.position.Y + this.sprite.Height < 0)
        this.position.Y = Asteroids.Screen.Y;
    else if (this.position.Y > Asteroids.Screen.Y)
        this.position.Y = -this.sprite.Height;
}
```

(d) *(5 punten)* Werk de header en body van de intersection methode uit. Je mag er vanuit gaan dat de rechthoeken die worden meegegeven als parameter altijd overlappen (dus je kunt altijd een overlappende rechthoek berekenen). Let op: deze methode moet voor alle mogelijke configuraties van overlapping werken!

Antwoord:

```
public static Rectangle intersection(Rectangle rect1, Rectangle rect2)
{
    int xmin = (int)MathHelper.Max(rect1.Left, rect2.Left);
    int xmax = (int)MathHelper.Min(rect1.Right, rect2.Right);
    int ymin = (int)MathHelper.Max(rect1.Top, rect2.Top);
    int ymax = (int)MathHelper.Min(rect1.Bottom, rect2.Bottom);
    return new Rectangle(xmin, ymin, xmax - xmin, ymax - ymin);
}
```

(e) *(7 punten)* De volgende stap is het toevoegen van een methode collidesWith aan de SpriteGameObject klasse die bepaalt of het huidige sprite game object botst met een ander sprite game object dat meegegeven wordt als parameter. Deze methode kijkt per pixel in het overlappende gebied of er een botsing is. Werk de header en body van deze methode uit. Gebruik hierbij de intersection en getPixelColor methode. Zorg ervoor dat de methode efficient is. Oftewel: er wordt alleen per-pixel collision checking gedaan als de bounding boxes van de twee objecten overlappen.

Antwoord:

```
public bool collidesWith(SpriteGameObject obj)
{
    if (!BoundingBox.Intersects(obj.BoundingBox))
        return false;
    Rectangle b = SpriteGameObject.intersection(BoundingBox, obj.BoundingBox);
    for (int x = 0; x < b.Width; x++)
        for (int y = 0; y < b.Height; y++)
        {
            if (getPixelColor(b.X − (int)position.X + x, b.Y − (int)position.Y + y).A != 0
                && obj.getPixelColor(b.X − (int)obj.position.X + x, b.Y − (int)obj.position.Y + y).A != 0)
                return true;
        }
    return false;
}
```

Ter informatie, de volledige getPixelColor methode is gegeven als volgt:

```
public Color getPixelColor(int x, int y)
{
    Rectangle sourceRectangle = new Rectangle(x, y, 1, 1);
    Color[] retrievedColor = new Color[1];
    sprite.GetData<Color>(0, sourceRectangle, retrievedColor, 0, 1);
    return retrievedColor[0];
}
```

(f) *(3 punten)* Tenslotte willen we onze per-pixel collision checking gebruiken in de game. We gaan dit doen in de Update methode van de Ufo klasse. Werk uit welke instructies aan die methode toegevoegd moeten worden zodat voor elke asteroide gekeken wordt of die botst met de ufo. Zo ja, dan moet de ufo rood worden. Zodra er geen botsing meer is, moet de ufo weer gewoon wit getekend worden. Let op dat *alleen* botsingen tussen de ufo en de asteroiden afgehandeld moeten worden. De andere game objecten (zoals de planeten en de achtergrondsprite) moeten genegeerd worden.

Antwoord:

```
this.color = Color.White;
foreach (SpriteGameObject g in Asteroids.GameWorld.GameObjects)
{
    if (g is Asteroid && g.collidesWith(this))
        color = Color.Red;
}
```