Derde deeltentamen **Gameprogrammeren**
Vrijdag 12 november 2010, 8.30-10.30 uur

- **Let op: Lever het tentamen in op twee bladen! Opgave 1 moet uitgewerkt worden op het eerste blad, opgave 2 op het tweede blad.** Schrijf op elk ingeleverd blad je naam en je studentnummer.
- De opgaven en de lijst met standaardfuncties mag je houden (behalve als je heel vroeg vertrekt).
- Het tentamen bestaat uit 2 opgaven. Elke opgave telt voor 50% mee. Als je een deel van een opgave niet weet, probeer dan toch zo veel mogelijk op te schrijven!

Veel succes!

---

1. Gegeven is de volgende klasse:

```
class TetrisBlok
{
    public TetrisBlok(Texture2D b)
    {
        this.kleur = Color.LightSalmon;
        this.blokSprite = b;
    }

    protected Color kleur;
    protected bool[,] configuratie;
    protected Texture2D blokSprite;

    private const int configuratieGrootte = 4;
    private const float blokGrootte = 30;

    public void Draw(Vector2 pos, SpriteBatch s)
    {
        // TODO
    }

    public void DraaiLinksom()
    {
        // TODO
    }

}
```

Deze klasse gebruiken we om een blok voor te stellen in de game Tetris. We gaan er in deze klasse vanuit dat de hoogte en breedte van één deelblokje 30 is (zie de variabele `blokGrootte`), en dat elk tetrisblok in een configuratie van 4x4 deelblokjes beschreven wordt, waarbij `true` staat voor bezet en `false` staat voor vrij. In de tweedimensionale array staat de eerste dimensie voor de x-positie en de tweede dimensie voor de y-positie van elke deelblokje in de configuratie.

(a) *(1 punt)* De tweedimensionale array is nog niet geinitialiseerd. Schrijf de opdrachten op waarmee we de array `configuratie` initialiseren en geef aan waar die opdrachten moeten staan. Standaard is een tetrisblok 'leeg', oftewel: alle posities in de array zijn vrij.

(b) *(1 punt)* We willen dat andere objecten kunnen opvragen of een bepaald element in de configuratie bezet is. Als de gegeven positie binnen de indices valt (oftewel $0 <= x < 4$ en $0 <= y < 4$), en de positie is bezet, dan en alleen dan is het resultaat `true`. Als de positie buiten de indices van de configuratie valt, dan is het resultaat altijd `false`. Moeten we hiervoor een methode of een property toevoegen aan de klasse? Schrijf de methode/property op die toegevoegd moet worden.

(c) *(2 punten)* Vul de `Draw`-methode in. Je mag er hierbij vanuit gaan dat de `blokSprite` een grootte van `blokGrootte * blokGrootte` heeft.

(d) *(3 punten)* Vul de `DraaiLinksom`-methode in. Deze methode draait het tetrisblok 90 graden naar links.

(e) *(1 punt)* Stel, we maken een subklasse van de klasse `TetrisBlok` genaamd `VierkantTetrisBlok`. Omdat het draaien van een vierkant blok weer hetzelfde blok oplevert willen we dat de `DraaiLinksom`-methode niks meer doet in de subklasse. We definiëren de klasse daarom als volgt:

```
class VierkantTetrisBlok : TetrisBlok
{
    public VierkantTetrisBlok(Texture2D b) : base(b)
    {
    }
    public void DraaiLinksom()
    {
        return;
    }
}
```

Dit heeft echter niet het gewenste effect. Wat gaat hier mis?

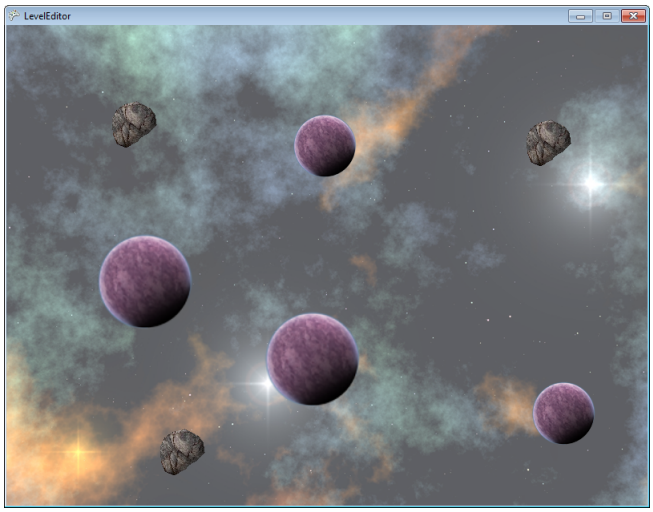
(f) *(2 punten)* Geef voor elk van de volgende expressies aan of hij gebruikt mag worden in de `DraaiLinksom`-methode van de klasse `VierkantTetrisBlok` (een 'ja' of een 'nee' is voldoende):

```
this.kleur
base.configuratie
this.configuratieGrootte
base.configuratieGrootte
```

2. In deze opgave gaan we een eenvoudige leveleditor schrijven voor een game die zich in de ruimte afspeelt. Een level bestaat uit asteroiden en planeten die op verschillende posities in een level staan. Een level wordt opgeslagen in een tekstbestand, waarbij we voor ieder object in het level het type object (asteroide of planeet), de x-positie, en de y-positie opslaan. Voor objecten van het type planeet bewaren we ook de massa, uitgedrukt in het aantal keer de massa van de aarde. Een voorbeeld van een level is gegeven als volgt:

```
asteroide 646 117
asteroide 129 94
planeet 299 350 2
planeet 90 253 2
planeet 342 106 1
planeet 640 440 1
asteroide 189 503
```

Na het laden ziet dit level er zo uit:



In de appendix van deze toets zijn al een aantal klassen gegeven. We hebben een klasse `LevelEditor`, waarin we de hoofdfunctionaliteit van de editor hebben geprogrammeerd. Na het laden van de sprites in de `LoadContent`-methode wordt de methode `leesUitBestand` aangeroepen waar we het level laden uit een tekstbestand. Aan het eind van het programma (als `game.Run()` klaar is), roepen we de methode `schrijfNaarBestand` aan, die het level weer bewaart in hetzelfde tekstbestand. Het level kan aangepast worden door de gebruiker door in het scherm te klikken. Als de gebruiker links klikt, dan wordt een asteroide toegevoegd aan het level op de positie waar de gebruiker klikt. Als de gebruiker de linkershifttoets ingedrukt houdt tijdens het klikken dan wordt een planeet met de massa van de aarde toegevoegd aan het level. Als de gebruiker de linkercontroltoets ingedrukt houdt, dan wordt een grote planeet (twee keer de massa van de aarde) toegevoegd. Dit gedrag is al uitgeprogrammeerd in de `Update`-methode.

Omdat we alles netjes object-georiënteerd op willen zetten hebben we een klasse `GameObject` en twee klassen `Planeet` en `Asteroide` die van `GameObject` erven.

(a) *(1 punt)* Tijdens het compileren van de klassen `Planeet` en `Asteroide` krijgen we compiler errors. Waarom? En wat moeten we doen om dit op te lossen? Je hoeft alleen maar aan te geven wat er moet gebeuren, je hoef dit niet uit te programmeren.

(b) *(2 punten)* In de `LevelEditor`-klasse mist nog een declaratie en een initialisatie. Schrijf deze twee opdrachten op en geef aan waar ze in de klasse moeten komen te staan.

(c) *(2 punten)* Maak de `Draw`-methode af zodat alle elementen in het level getekend worden.

(d) *(2 punten)* Maak de `schrijfNaarBestand`-methode af. Maak hierbij gebruik van de `NaarString`-methode uit de `GameObject`-klasse. De string die het pad van het tekstbestand bevat is al gegeven (`levelPad`).

(e) *(3 punten)* Maak de `leesUitBestand`-methode af. De string die het pad van het tekstbestand bevat is al gegeven (`levelPad`).

GAME OVER

Appendix: klassen

LevelEditor

```
class LevelEditor : Game
{
    SpriteBatch spriteBatch;
    Texture2D achtergrondSprite, planeetSprite, asteroideSprite;
    MouseState vorigeMuisStatus;

    static void Main()
    {
        LevelEditor game = new LevelEditor();
        game.Run();
        game.schrijfNaarBestand();
    }

    public LevelEditor()
    {
        GraphicsDeviceManager graphics = new GraphicsDeviceManager(this);
        Content.RootDirectory = "Content";
        this.IsMouseVisible = true;
    }
    public void schrijfNaarBestand()
    {
        string levelPad = Path.Combine(StorageContainer.TitleLocation, "Content/level.txt");
        // TODO: schrijf level naar bestand
    }
    public void leesUitBestand()
    {
        string levelPad = Path.Combine(StorageContainer.TitleLocation, "Content/level.txt");
        if (!File.Exists(levelPad))
            return;
        // TODO: lees level uit bestand
    }
    protected override void LoadContent()
    {
        spriteBatch = new SpriteBatch(GraphicsDevice);
        this.achtergrondSprite = Content.Load<Texture2D>("achtergrond");
        this.asteroideSprite = Content.Load<Texture2D>("asteroide");
        this.planeetSprite = Content.Load<Texture2D>("planeet");
        this.leesUitBestand();
    }
    protected override void Update(GameTime gameTime)
    {
        MouseState huidigeMuisStatus = Mouse.GetState();
        KeyboardState toetsenbordStatus = Keyboard.GetState();
        if (huidigeMuisStatus.LeftButton == ButtonState.Pressed &&
            vorigeMuisStatus.LeftButton == ButtonState.Released)
        {
            Vector2 positie = new Vector2(huidigeMuisStatus.X, huidigeMuisStatus.Y);
            if (toetsenbordStatus.IsKeyDown(Keys.LeftShift))
                levelObjecten.Add(new Planeet(planeetSprite, positie, 1));
            else if (toetsenbordStatus.IsKeyDown(Keys.LeftControl))
                levelObjecten.Add(new Planeet(planeetSprite, positie, 2));
            else
                levelObjecten.Add(new Asteroide(asteroideSprite, positie));

        }
        vorigeMuisStatus = huidigeMuisStatus;
    }
```

```
    protected override void Draw(GameTime gameTime)
    {
        spriteBatch.Begin();
        spriteBatch.Draw(this.achtergrondSprite, Vector2.Zero, Color.White);
        // TODO: teken level objecten
        spriteBatch.End();
    }
}
```

GameObject

```
abstract class GameObject
{
    protected Texture2D sprite;
    protected Vector2 positie;

    public GameObject(Texture2D geladenSprite, Vector2 pos)
    {
        this.sprite = geladenSprite;
        this.positie = pos;
    }

    public abstract string NaarString();

    public virtual void Draw(SpriteBatch spriteBatch)
    {
        spriteBatch.Draw(this.sprite, this.positie, Color.White);
    }
}
```

Asteroide

```
class Asteroide : GameObject
{
    public Asteroide(Texture2D geladenSprite, Vector2 pos)
        : base(geladenSprite, pos)
    {
    }
}
```

Planeet

```
class Planeet : GameObject
{
    int massa;

    public Planeet(Texture2D geladenSprite, Vector2 pos, int massa)
        : base(geladenSprite, pos)
    {
        this.massa = massa;
    }

    public override void Draw(SpriteBatch spriteBatch)
    {
        spriteBatch.Draw(this.sprite, this.positie, null, Color.White, 0.0f,
            Vector2.Zero, this.massa, SpriteEffects.None, 0.0f);
    }
}
```