Eerste deeltentamen Gameprogrammeren
Vrijdag 28 september 2012, 11.00-13.00 uur

Naam:	
Studentnummer:	

- Het tentamen bestaat uit 4 opgaven. Elke opgave levert 10 punten op. Je cijfer is het totaal aantal punten gedeeld door 4. Als je een deel van een opgave niet weet, probeer dan toch zo veel mogelijk op te schrijven!
- De lijst met standaardfuncties na afloop graag weer inleveren.

Veel succes!

---

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

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

   - ☐ Elke klasse moet tenminste één methode hebben die de naam Main draagt.
   - ☐ Er bestaan methoden en properties die geen objecten bewerken.
   - ☐ Als het woord **new** gebruikt wordt om een object te creëren, dan weten we dat dat object als type een klasse heeft.
   - ☐ Het woord **this** mag niet gebruikt worden in een **static** methode.
   - ☐ Elke **while** opdracht kan omgeschreven worden naar een **for** opdracht en vice versa.
   - ☐ Scripttalen zijn altijd imperatief.

   (b) *(2 punten)* De vertaling van een C#-programma naar machinecode gebeurt in twee stappen. Eerst wordt het programma vertaald naar een intermediate language, en die wordt vervolgens weer vertaald naar machinecode. Noem twee voordelen van het vertalen in twee stappen.

   1:

   2:

(c) *(3 punten)* Beschouw de volgende methode:

```
public int deelsom(int n)
{
    int resultaat = 0;
    for (int i = n; i >= 0; i -= 2)
        resultaat += i;
    return resultaat;
}
```

Geef aan wat de uitkomst is van de volgende expressies:

deelsom(8):

deelsom(5):

deelsom(1):

deelsom(−1):

Schrijf de **for** loop om naar een **while** loop.

(d) *(2 punten)* Wat betekent het als het woord **void** voor de naam van een methode in diens methodeheader wordt geschreven? Is de aanroep van die methode dan een expressie of een opdracht? Of allebei? Of geen van beiden?

2. *($-\frac{1}{2}$ punt per fout)* Hieronder staat 16 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

(**int**)		'\\'		**decimal**		**for**(;**false**;);	
int i;		Texture2D.Height;		Color.White		**true**==**false**	
Texture2D t;		**new** SpriteBatch(GraphicsDevice)		1 + 2 = 3		1E1	
3==i<5		Vector2.Zero / 2		SpriteBatch		/*1 */2 //3	

Opgave 3 en 4 vragen een stukje programma. Kleine schrijffoutjes (hoofdletters, puntkomma's enz.) worden niet streng afgerekend, maar de elementen die de structuur van het programma bepalen (haakjes, accolades, aanhalingstekens enz.) zijn wel belangrijk. Schrijf die dus duidelijk en op de goede plaats op! Het is toegestaan (maar niet nodig) om C#-constructies die (nog) niet zijn behandeld toch te gebruiken. Je hoeft niet aan te geven welke **using***-opdrachten nodig zijn om de klassen te kunnen gebruiken.*

3. Als de som van de alle cijfers tot de macht 3 van een getal hetzelfde is als dat getal, dan noemen we het getal ook wel een *Armstronggetal*. Bijvoorbeeld, 153 is een Armstronggetal, want:

$$153 = (1 \times 1 \times 1) + (5 \times 5 \times 5) + (3 \times 3 \times 3)$$

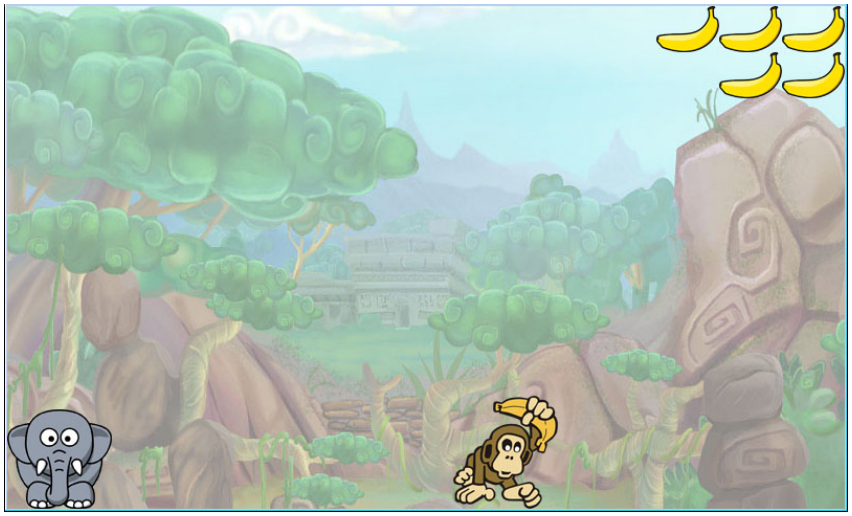
   (a) *(5 punten)* Schrijf een statische methode isArmstrongNumber die van een gegeven getal bepaalt of het een Armstronggetal is. Je mag er hierbij vanuit gaan dat altijd een positief getal meegegeven wordt aan de methode.

(b) *(3 punten)* Schrijf een statische methode isPrimeNumber die uitrekent of een gegeven getal een priemgetal is. Een priemgetal is een getal dat alleen deelbaar door één en door zichzelf is. Ook hier mag je er vanuit gaan dat altijd een positief getal meegegeven wordt aan de methode.

(c) *(2 punten)* Schrijf een serie opdrachten die alle getallen tussen 1 en 500 (eenmalig) op het scherm afdrukt die een Armstronggetal of een priemgetal zijn, of beide. Gebruik de methoden uit de vorige deelopdrachten. Je mag er vanuit gaan dat deze opdrachten in een console applicatie uitgevoerd worden.

4. De maat is vol! De olifanten hebben er genoeg van dat de apen steeds hun bananen stelen en opeten. Ze hebben geprobeerd er met de apen over te praten, maar het enige antwoord dat ze kregen was "oeh oeh aah aah". De enige oplossing die overblijft is bovenop die vervelende apen te springen, dat zal ze leren!

   En zo begint het spel "Annoyed Elephants". Het doel van dit spel is om op zoveel mogelijk apen te springen. Dit doe je door in het scherm te klikken met je muis, waardoor de olifant springt. De plek waar je klikt bepaalt de snelheid en richting van de sprong. Elke keer als je op een aap springt, krijg je een banaan. De aap wordt dan op een andere willekeurige plek onderaan het scherm geplaatst en de olifant staat weer links onderin. Het aantal bananen dat je tot nu toe hebt verzameld wordt rechtsbovenin het scherm afgebeeld. In de appendix van deze toets staan de belangrijkste klassen die gebruikt worden in deze game. En dit is een screenshot van het spel:



   (a) *(2 punten)* Wat is de reden dat sommige membervariabelen in de AnnoyedElephants klasse **static** zijn?

Het aantal bananen dat je als speler verzameld hebt staat in de membervariabele points in de GameWorld klasse. Zoals je kunt zien in de screenshot worden er maximaal drie bananen naast elkaar getekend. Als je dus als speler bijvoorbeeld tot nu toe vijf bananen verzameld hebt, dan worden er twee rijen bananen getekend: één rij met drie bananen, en daaronder een rij met twee bananen.

(b) *(4 punten)* Het tekenen van de bananen gebeurt in de methode DrawBananas uit de GameWorld klasse. Werk deze methode uit (header en body). Let erop dat de bananen altijd netjes helemaal rechts in het scherm getekend worden, ook als de speler minder dan drie bananen verzameld heeft.

(c) *(2 punten)* Er mist nog een property Points in de GameWorld klasse. Werk deze property uit. Zorg ervoor dat het aantal verkregen punten nooit kleiner kan zijn dan nul.

(d) *(2 punten)* Als een olifant botst met een aap, dan wordt de aap op een willekeurige plek onderaan het scherm geplaatst. Echter, de x-positie van de aap mag nooit kleiner zijn dan 100, omdat de aap dan op de positie van de olifant zou kunnen komen. Het plaatsen van de aap gebeurt in de Reset methode. Werk de body van deze methode uit.

Appendix: klassen

AnnoyedElephants

```
class AnnoyedElephants : Game
{
    GraphicsDeviceManager graphics;
    SpriteBatch spriteBatch;
    InputHelper inputHelper;
    static Random random;
    static GameWorld gameWorld;
    static Vector2 screen;

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

    public AnnoyedElephants() {
        Content.RootDirectory = "Content";
        IsMouseVisible = true;
        graphics = new GraphicsDeviceManager(this);
        random = new Random();
        inputHelper = new InputHelper();
    }

    protected override void LoadContent() {
        spriteBatch = new SpriteBatch(GraphicsDevice);
        screen = new Vector2(GraphicsDevice.Viewport.Width, GraphicsDevice.Viewport.Height);
        gameWorld = new GameWorld(Content);
    }

    protected override void Update(GameTime gameTime) {
        inputHelper.Update();
        gameWorld.HandleInput(inputHelper);
        gameWorld.Update(gameTime);
    }

    protected override void Draw(GameTime gameTime) {
        GraphicsDevice.Clear(Color.White);
        spriteBatch.Begin();
        gameWorld.Draw(gameTime, spriteBatch);
        spriteBatch.End();
    }

    public static Vector2 Screen {
        get { return screen; }
    }

    public static Random Random {
        get { return random; }
    }

    public static GameWorld GameWorld {
        get { return gameWorld; }
    }
}
```

GameWorld

```
class GameWorld
{
    Texture2D background, banana;
    Elephant elephant;
    Monkey monkey;
    int points;

    public GameWorld(ContentManager Content)
    {
        background = Content.Load<Texture2D>("background");
        banana = Content.Load<Texture2D>("banana");
        elephant = new Elephant(Content);
        monkey = new Monkey(Content);
        points = 0;
    }

    public void HandleInput(InputHelper inputHelper)
    {
        elephant.HandleInput(inputHelper);
    }

    public void Update(GameTime gameTime)
    {
        elephant.Update(gameTime);
        monkey.Update(gameTime);
    }

    public void Draw(GameTime gameTime, SpriteBatch spriteBatch)
    {
        spriteBatch.Draw(background, Vector2.Zero, Color.White);
        DrawBananas(spriteBatch);
        monkey.Draw(gameTime, spriteBatch);
        elephant.Draw(gameTime, spriteBatch);
    }

    public bool IsOutsideWorld(Vector2 position)
    {
        return position.X < 0 || position.X > AnnoyedElephants.Screen.X || position.Y > AnnoyedElephants.Screen.Y;
    }

    public Elephant Elephant
    {
        get { return elephant; }
    }

    public Monkey Monkey
    {
        get { return monkey; }
    }
}
```

Elephant

```
class Elephant
{
    Texture2D sprite;
    Vector2 position, velocity;

    public Elephant(ContentManager Content)
    {
        sprite = Content.Load<Texture2D>("elephant");
        Reset();
    }

    public void HandleInput(InputHelper inputHelper)
    {
        if (inputHelper.MouseLeftButtonPressed() && velocity == Vector2.Zero)
            velocity = (inputHelper.MousePosition - position) / 40;
    }

    public void Update(GameTime gameTime)
    {
        if (velocity != Vector2.Zero)
        {
            velocity.X *= 0.99f;
            velocity.Y += 0.1f;
        }
        position += velocity;
        if (AnnoyedElephants.GameWorld.IsOutsideWorld(position))
            Reset();
    }

    public void Reset()
    {
        position = new Vector2(0, AnnoyedElephants.Screen.Y - sprite.Height);
        velocity = Vector2.Zero;
    }

    public Vector2 CenterPosition
    {
        get { return position + new Vector2(sprite.Width, sprite.Height) / 2; }
    }

    public void Draw(GameTime gameTime, SpriteBatch spriteBatch)
    {
        spriteBatch.Draw(sprite, position, Color.White);
    }
}
```

Monkey

```
class Monkey
{
    Texture2D sprite;
    Vector2 position, velocity;

    public Monkey(ContentManager Content)
    {
        sprite = Content.Load<Texture2D>("monkey");
        Reset();
    }

    public void Update(GameTime gameTime)
    {
        Vector2 distanceVector = AnnoyedElephants.GameWorld.Elephant.CenterPosition - CenterPosition;
        if (distanceVector.Length() < 100)
        {
            AnnoyedElephants.GameWorld.Points++;
            AnnoyedElephants.GameWorld.Elephant.Reset();
            Reset();
        }
    }

    public void Reset()
    {
        // TODO
    }

    public Vector2 CenterPosition
    {
        get { return position + new Vector2(sprite.Width, sprite.Height) / 2; }
    }

    public void Draw(GameTime gameTime, SpriteBatch spriteBatch)
    {
        spriteBatch.Draw(sprite, position, Color.White);
    }
}
```

EINDE TOETS