Uitwerkingen derde deeltentamen Gameprogrammeren
Vrijdag 8 november 2013, 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) *(1.5 punt)* Wat is een edit state? Waarvoor wordt een edit state gebruikt in games?

   Antwoord: Een edit state is een game state die gebruik maakt van een andere game state. Een voorbeeld is een 'level finished' toestand die de gamewereld laat zien (en dus een 'playing state' gebruikt), maar die daar bovenop een overlay tekent.

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

   - ☐ Elke **if**-opdracht kan worden omgeschreven naar een **switch** opdracht en vice versa.
   - X Excepties gebruik je vooral om het programma te stoppen in het geval dat er een fout optreedt die niet door de programmeur voorzien had kunnen worden, zoals een missend tekstbestand, of een verstoorde netwerkverbinding.
   - ☐ Een abstracte klasse is een klasse die geen membervariabelen, en implementaties van properties en methoden bevat.
   - X Een klasse mag meerdere interfaces implementeren, maar mag niet erven van meerdere abstracte klassen.
   - X Objecten van het type **string** zijn immutable.
   - ☐ Botsingen tussen game objecten worden afgehandeld in de Draw methode, want we weten pas of objecten botsen nadat ze getekend zijn.

   (c) *(1.5 punt)* Voor wat voor soort fouten worden excepties gebruikt in een programma? Geef een concreet voorbeeld binnen de context van games.

   Antwoord: Excepties worden geworpen voor gevallen waarin er een (extern) probleem is waar tijdens het compileren niet op gecontroleerd kan worden. Een voorbeeld hiervan is het plotseling wegvallen van een internetverbinding bij een multiplayer game, of game assets die missen.

   (d) *(4 punten)* Hieronder staan vier programmafragmenten. Geef aan wat de waarde van de variable teller is na het uitvoeren van elk programmafragment.

1	**int** teller = 0;
	for (**int** i = 0; i < 10; i++)
	{
	  if (i % 3 == 0)
	  continue;
	  teller++;
	}
		6

2	**int** teller = 0;
	for (**int** i = 0; i < 10; i++)
	{
	  if (i % 3 == 0)
	  break;
	  teller++;
	}
		0

3	**int** teller = 0;
	while (teller < 10)
	  for (**int** i = 1; i <= 10; i++)
	  {
	  if (i % 4 == 0)
	  break;
	  teller++;
	  }
		12

4	**int** teller = 0;
	for (**int** i = 0; i <= 5; i++)
	  switch (i)
	  {
	  case 1: teller++; **break**;
	  case 2:
	  case 3: teller += 2; **break**;
	  case 4: teller−−; **break**;
	  default: teller ∗= 2; **break**;
	  }
		8

2. (a) *(5 punten)* Schrijf de methode Replace.

Antwoord:

```
string Replace(char oud, char nieuw)
{
    string res = "";
    for (int t=0; t<this.Length; t++)
    {
        if (this[t]==oud)
            res += nieuw;
        else res += this[t];
    }
    return res;
}
```

(b) *(5 punten)* Schrijf een statische methode met als parameter een array met strings, die de langste string van de array oplevert. Je mag aannemen dat de array minstens één string bevat. Als er meerdere strings zijn die even lang zijn mag je zelf kiezen welke je daarvan oplevert.

Antwoord:

```
static string Langste(string[] a)
{
    string res = "";
    for (int t=0; t<a.Length; t++)
        if (a[t].Length > res.Length)
            res = a[t];
    return res;
}
```

3. (a) *(4 punten)* De methode LaadTiles in de Level klasse roept een methode LaadTile aan die een tile bouwt aan de hand van het gelezen karakter. Deze methode zorgt er ook voor dat de lijst met dozen gevuld wordt en dat het speler object gemaakt wordt. Werk deze methode uit met behulp van de **switch** opdracht. Daarnaast moet, als deze methode een tile karakter tegenkomt dat ongeldig is, een *exceptie* geworpen worden.

Antwoord:

```
private Tile LaadTile(char tileType, int x, int y)
{
    switch (tileType)
    {
        case '.':
            return new Tile(null, TileType.Achtergrond, x, y);
        case 'W':
            return new Tile(muurSprite, TileType.Muur, x, y);
        case 'B':
            dozen.Add(new Tile(doosSprite, TileType.Doos, x, y));
            return new Tile(null, TileType.Achtergrond, x, y);
        case 'G':
            return new Tile(doelSprite, TileType.Doel, x, y);
        case 'P':
            speler = new Tile(spelerSprite, TileType.Speler, x, y);
            return new Tile(null, TileType.Achtergrond, x, y);
        default:
            throw new NotSupportedException("Onbekend tile type: " + tileType);
    }
}
```

(b) *(2 punten)* Om makkelijk uit te kunnen vinden of een doos zich bevindt op een bepaalde (x, y)-positie willen we ook een methode VindDoos maken in de Level klasse. Deze methode krijgt als parameter een Vector2 object mee en die loopt door de lijst van dozen om te zien of een van die dozen op de gevraagde positie staat. Zo ja, dan wordt die doos als resultaat opgeleverd. Zo nee, dan levert de methode de waarde **null** op. Implementeer deze methode.
Let op: het gaat hier niet om de pixel-coördinaten, maar om de locatie in het tweedimensionale tileveld!

Antwoord:

```
public Tile VindDoos(Vector2 pos)
{
    foreach (Tile t in dozen)
        if (t.Pos == pos)
            return t;
    return null;
}
```

(c) **Antwoord:**

```
private bool IsGeldigePlaats(Vector2 pos)
{
    if (pos.X < 0 || pos.X >= tiles.GetLength(0) || pos.Y < 0 || pos.Y >= tiles.GetLength(1))
        return false;
    if (tiles[(int)pos.X, (int)pos.Y].TileType == TileType.Muur || VindDoos(pos) != null)
        return false;
    return true;
}
```

(d) **Antwoord:**

```
Vector2 offset = Vector2.Zero;
if (inputHelper.KeyPressed(Keys.Left))
    offset.X−−;
else if (inputHelper.KeyPressed(Keys.Right))
    offset.X++;
else if (inputHelper.KeyPressed(Keys.Up))
    offset.Y−−;
else if (inputHelper.KeyPressed(Keys.Down))
    offset.Y++;
else
    return;
Vector2 spelerdoel = speler.Pos + offset;
Vector2 doosdoel = spelerdoel + offset;

Tile doos = VindDoos(spelerdoel);
if (doos != null && IsGeldigePlaats(doosdoel))
    doos.Pos = doosdoel;
if (IsGeldigePlaats(spelerdoel))
    speler.Pos = spelerdoel;
```

(e) *(3 punten)* Een level is afgemaakt als alle dozen op een doelpositie staan. Implementeer een alleen-lezen property LevelAfgemaakt die aangeeft of een level afgemaakt is.
Antwoord:

```
public bool LevelAfgemaakt
{
    get
    {
        foreach (Tile b in dozen)
            if (tiles[(int)b.Pos.X, (int)b.Pos.Y].TileType != TileType.Doel)
                return false;
        return true;
    }
}
```

(f) *(1 punt)* Is het een goede ontwerpkeuze geweest om de Tile klasse ook te gebruiken voor de speler en de dozen? Zo ja, wat zijn de nadelen om het anders te doen? Zo nee, wat zou een betere keuze zijn geweest?
Antwoord:
Het is eigenlijk geen goede ontwerpkeuze omdat op deze manier de x en y-posities dubbel worden opgeslagen van de achtergrond, muur en doeltiles. Ook zijn de speler en de dozen in principe geen tiles. Een betere keuze zou zijn geweest om een algemene GameObject klasse te maken, waarvan een Tile erft. We kunnen dan vervolgens subklassen maken van die GameObject klasse.