SDL LIBRARY: UITBREIDINGEN

PLATFORM GAME: INDELEN VAN JE LEVEL

Bij platform games of ook bij andere spelen, moet je trachten op een makkelijke manier je level in te delen. Bijvoorbeeld op positie (x,y) staan stenen waar je spelfiguur moet op springen, iets hoger staan nog een paar stenen gedefinieerd , verder naar links ga je een vijand plaatsen enz.. Dit hoofdstuk geeft je een mogelijke oplossing over hoe je je level kan opbouwen.

Om mijn objecten (obstakels, vijanden, …) ergens in de level te plaatsen gebruik ik 2 arrays:

ARRAY1

Onderstaande array bepaalt de plaats van een object: de array is 50x20 dimensies groot, en dit alles doe je maal 30, omdat elk object een veelvoud van 30 pixels is. Dus met andere woorden heb ik een level opgebouwd dat 50x30 = 1500 pixels breed en 20x30=600 pixels hoog. Onderstaande array fungeert als plattegrond van je level.

```
public byte[,] intTileArray = new byte[,] // elk getal > 0 stelt een bepaald blokje voor
{
    {0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    {0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    {0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    {0,2,2,0,0,0,0,0,2,2,2,2,0,0,0,0,1,0,0,2,2,2,2,2,2,2,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    {2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
};
```

ARRAY2

Vervolgens implementeer je nu een tweede array zodat op de juiste plaats de juiste objecten kunnen instantieren. (bijvoorbeeld cijfer 2 staat voor gras, cijfer 1 staat voor een baksteen, …). Deze array kan er bij voorbeeld zo uit zien:

```
private void CreateWorld()
        {
            //read map and place the right objects in the world


            for (int x = 0; x <= 19; x++)
            {
                for (int y = 0; y <= 52; y++)
                {
                    switch (intTileArray[x, y])
                    {
                        case 1:
                           spriteTileArray[x, y] = new grass("grass.png", y * 30, x *
                           30, theVideo);
                            break;

                        case 2:
                           spriteTileArray[x, y] = new Brick("brick.png", y * 30, x *
                           30, theVideo);
                            break;

                        case 3:
                           spriteTileArray[x, y] = new Turtle("turtle.png", y * 30, x
                           * 30, theVideo);
                            break;

                        case 4:
                           spriteTileArray[x, y] = new Boss("kong.png", y * 30, x *
                           30, theVideo);
                            break;
                    }
                }
            }
        }
```

Je merkt hierboven onmiddellijk het polymorfe gedrag van de objecten (n die in 1 lijst kunnen worden geplaatst!)

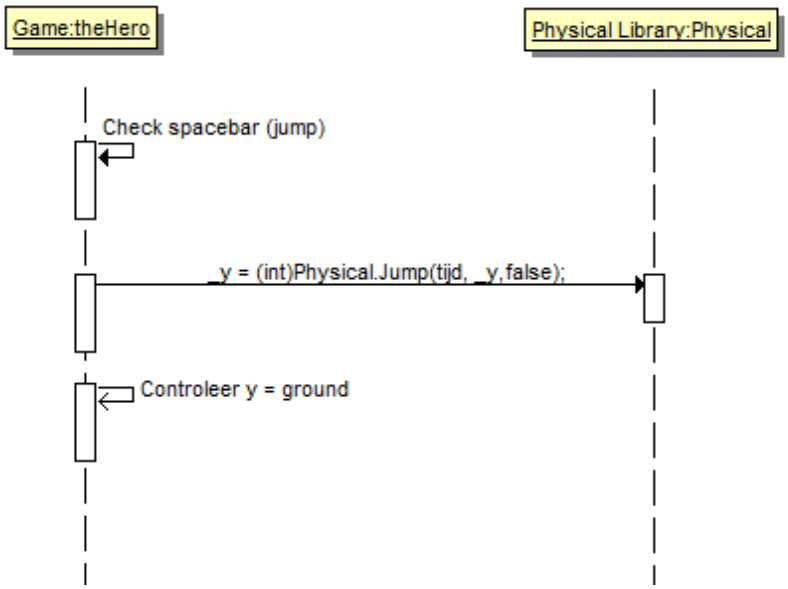
Een tweede stap zou het tekenen van de betreffende objecten zijn. Dit kan als volgt gebeuren:

```
public void DrawLevelItems()
        {


            for (int x = 0; x < 8; x++)
            {
                for (int y = 0; y < 8; y++)
                {
                    if (_currentLevel.spriteTileArray[x, y] != null)
                    {
                            _currentLevel.spriteTileArray[x, y].Draw();
                    }
                }
            }
        }
```

De spriteTileArray is een array van bijvoorbeeld Blok (dit is de hoofdklasse van al je afgeleide klasses die je in je gameworld tekent!

```
public Blok[,] spriteTileArray = new Blok[8, 8];  //De sprite array van mijn level
```

PHYSICAL LIBRARY



Functie Jump in de fysische bibliotheek:

```
public static double Jump(int tijd, int lastYPos,bool highJump)
        {
            double value = (highJump == true) ? 2.4 : 1.9;
              int result = -((int)((value * Math.Sin(0.8) * tijd) - (.5 * 0.1 * (tijd *
              tijd)))) + lastYPos;

               return result;
        }
```

In het TICK-event van de theHero klasse:

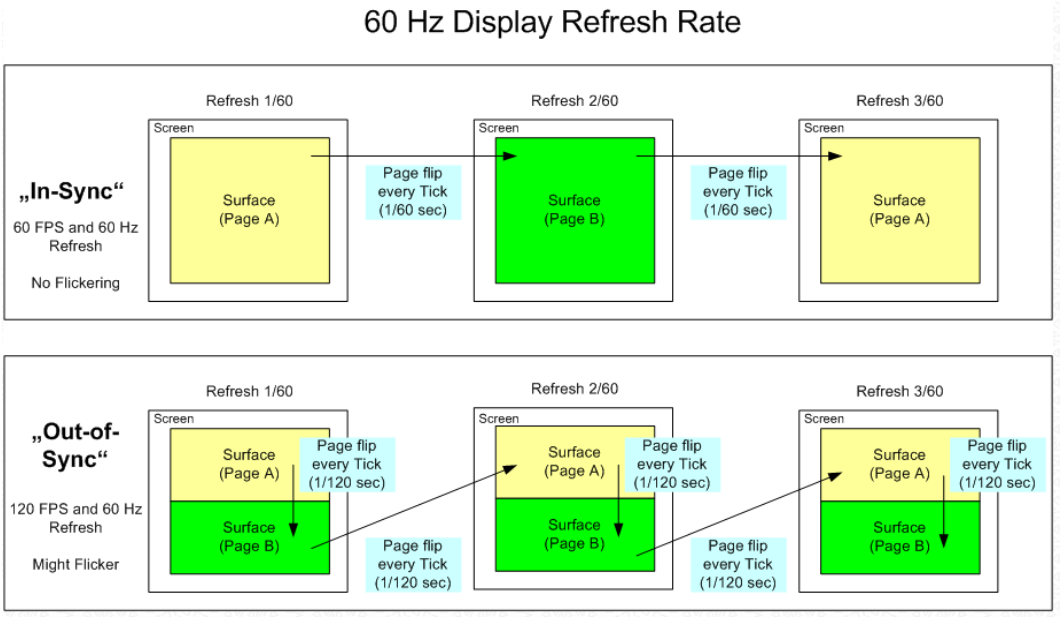
```
            if (_jump)
            {

                _y = (int)Physical.Jump(tijd, _y,false);
                tijd++;

                if (_y >= _ground)
                {
                    _jump = false;
                    //hack
                    _y = _ground;
                    tijd = 1;

                }
            }
```

FRAME INDEPENDENT MOVEMENT

Normaal gesproken staat is de TICK snelheid gelijk aan je scherm refresh snelheid (60 Hz). Met andere woorden: de Tick Event Handler wordt 60x per seconde afgehandeld. Je kan deze snelheid veranderen via de SDL bibliotheek door:

- Events.TargetFps : dit is de framerate die je wenst.
- Events.Fps: dit is de actuele frame rate.

Frame rate of scherm refresh rate aanpassen is niet aan te raden omwille van flickering problemen (als je screen refresh vraagt en dit valt tussen de refresh rate van je scherm..)



Als je sprites gaat bewegen zal het kunnen zijn dat je spel sneller gaat dan je wil.., de oplossing voor dit probleem is door gebruik te maken van **Frame Independent Movement.**

1. Als je je sprite tegen 200 pixels per seconde wil laten bewegen

Indien je de framerate op 2 frames per seconde zet, met andere woorden elke halve seconde wordt je tick event uitgevoerd,zal je dus tegen 100 pixels per tick moeten bewegen. Nu die halve seconde is ook niet helemaal waar, want als bijvoorbeeld je applicatie op 60 FPS draait, zal je TICK event elke 16,6ms (1000/60) verschijnen. Veronderstel dat je tick-handler er 5ms over doet om je code uit te voeren, rest er nog 11,6 ms tot de volgende tick. Het is dit verschil dat je in rekening moet brengen (dat verschil kan je in code opvragen door

:

```
void Events_Tick(object sender, TickEventArgs e)
        {

            Int tijd = e.TicksElapsed;

        }
```

IN CODE WORDT DE BEREKENING VOOR DE HOEVEELHEID PIXELS DIE JE WIL LATEN BEWEGEN PER TICK DUS : 1/1000*TICKSSINCELASTTICKEVENT* MOVEMENTPERSECONDSINPIXELS

Stel een op zich zelfstaande sprite die 60 x per seconde beweegt:

	
De applicatie	De sprite

De aap zal in je applicatie elke Tick een ander figuurtje laten zien, wat veel te snel is, daarom is het in dergelijke situatie aan te raden dat je een frame onafhankelijke snelheid kan voorzien.

Onderstaande code geeft weer hoe je frame onafhankelijk kan werken:

```
    class FrameRate
    {
        Surface video;
        Surface aap;

        public FrameRate()
        {
            video = Video.SetVideoMode(500, 500);
            aap = new Surface("kong.png");


            Events.Tick += new EventHandler<TickEventArgs>(Events_Tick);
            Events.Run();

        }

        int x = 0;
        int offset = 0;
        void Events_Tick(object sender, TickEventArgs e)
```

```
        {

            video.Fill(Color.Black);

            //beweeg 63 pixels per seconde
            double temp = (e.TicksElapsed * 63 )/1000;
            x += (int)temp;

            if (offset >= 378)
                offset = 0;

            if (x % 63 == 0)
            {
                offset += 63;
            }



            video.Blit(aap, new Point(250, 250), new Rectangle(offset, 0, 63, 60));
            video.Update();

        }
    }
}
```

STRATEGY PATTERN

Indien geen strategy patroon geïmplementeerd is, ga je snel in statische code hervallen. Zie onderstaand voorbeeld:

```
enum eLevel
    {
        level1,
        level2,
        level3
    }

    public class Strategy
    {
        private eLevel m_Level;
        Surface video;


        public Strategy()
        {
            video = Video.SetVideoMode(500, 500);
            m_Level = eLevel.level1;

            level1 = new Level1(video);
            level2 = new Level2(video);
            level3 = new Level3(video);


            Events.MouseButtonDown += new
EventHandler<SdlDotNet.Input.MouseButtonEventArgs>(Events_MouseButtonDown);
            Events.Tick += new EventHandler<TickEventArgs>(Events_Tick);
            Events.Run();


        }

        int klik = 1;
        void Events_MouseButtonDown(object sender,
SdlDotNet.Input.MouseButtonEventArgs e)
        {
            klik++;
            switch (klik)
            {
                case 1:
                    m_Level = eLevel.level1;
                    break;
                case 2:
                    m_Level = eLevel.level2;

                    break;
                case 3:
                    m_Level = eLevel.level3;
                    currentLevel = level3;
                    break;
            }
```

```
            klik = (klik == 3) ? 0 : klik;


        }

        void Events_Tick(object sender, TickEventArgs e)
        {

            switch (m_Level)
            {
                case eLevel.level1:
                    Action("level1");
                    break;
                case eLevel.level2:
                    Action("level2");
                    break;
                case eLevel.level3:
                    Action("level3");
                    break;
            }


            video.Fill(Color.Black);
            video.Update();

        }


        Surface fontSurface;
        private void Action(string l)
        {

            video.Fill(Color.Black);
            SdlDotNet.Graphics.Font font = new SdlDotNet.Graphics.Font(@"Arial.ttf",
42);
             //Create the Font Surfaces
            fontSurface = font.Render(l, Color.White);

            video.Blit(fontSurface);

            video.Update();


        }
```

UITLEG

Je bemerkt dat er een enum structuur gedefinieerd is, die de huidige level in leesbare taal weergeeft. In de klasse wordt een variabele aangemaakt van het enum type, die in de loop van de applicatie wijzigt naar level 1, level2 of level3. Afhankelijk van deze level wordt er een procedure opgeroepen die de juiste taken zal uitvoeren afhankelijk van het level waarin je je bevindt. Dit wordt zeer statische code, omdat je in de Action methode veel moet verwerken. Dus naar onderhoudbaarheid , editeerbaarheid misschien niet de juiste wijze van coderen. (hoewel dit wel perfect zal werken!).

Beter is om het Strategy patroon toe te passen, zie code hieronder:

STRATEGY PATTERN

```
public class Strategy
    {

        Surface video;

        private Level1 level1;
        private Level2 level2;
        private Level3 level3;
        private Level currentLevel;


        public Strategy()
        {
            video = Video.SetVideoMode(500, 500);

            level1 = new Level1(video);
            level2 = new Level2(video);
            level3 = new Level3(video);
            currentLevel = new Level1(video);


            Events.MouseButtonDown += new
EventHandler<SdlDotNet.Input.MouseButtonEventArgs>(Events_MouseButtonDown);
            Events.Tick += new EventHandler<TickEventArgs>(Events_Tick);
            Events.Run();


        }

        int klik = 1;
        void Events_MouseButtonDown(object sender,
SdlDotNet.Input.MouseButtonEventArgs e)
        {
            klik++;
            switch (klik)
            {
                case 1:

                    currentLevel = level1;
                    break;
                case 2:

                    currentLevel = level2;
                    break;
                case 3:

                    currentLevel = level3;
                    break;
            }

            klik = (klik == 3) ? 0 : klik;


        }

        void Events_Tick(object sender, TickEventArgs e)
        {
            //strategy pattern:
```

```
            video.Fill(Color.Black);
            currentLevel.Action();
            video.Update();

        }


        Surface fontSurface;
        private void Action(string l)
        {

            video.Fill(Color.Black);
            SdlDotNet.Graphics.Font font = new SdlDotNet.Graphics.Font(@"Arial.ttf",
42);
             //Create the Font Surfaces
            fontSurface = font.Render(l, Color.White);

            video.Blit(fontSurface);

            video.Update();


        }
```

Met als abstracte klasse “Level”:

```
  public abstract class Level
    {
        protected Surface m_Video;
        protected Surface fontSurface;
        protected SdlDotNet.Graphics.Font font;
        public Level(Surface video)
        {
            m_Video = video;

             font = new SdlDotNet.Graphics.Font(@"Arial.ttf", 42);
            // Create the Font Surfaces


        }
        public abstract void Action();
    }
}
```

En afgeleide klassen Level1, Level2, Level3:

Vb. Level1:

```
public class Level1 : Level
    {
        public Level1(Surface video) : base(video) { }

        public override void Action()
        {
            fontSurface = font.Render("Level1", Color.White);
            m_Video.Blit(fontSurface);
```

```
        }
    }
```