SDL LIBRARY

AUDIO

AFSPELEN VAN GELUID EN EFFECTEN

In elke game-applicatie word je geconfronteerd met geluidseffecten en achtergrondmuziek. In SDL.NET heb je de keuze hoe geluid wordt afgespeeld:

- Sound Class: voor effecten

- Music/MusicPlayer Class: voor langere samples zoals achtergrondmuziek.

SDL kan verschillende formaten zoals ogg, wav, mod, mid... laden.

Laten we onmiddellijk een voorbeeld project maken:

- Creëer een SDL window dat toont welk geluid afgespeeld wordt.
- Speel een langer achtergrond geluidsfragment af.
- Speel 3 samples in een loop af

PROJECT AUDIO

```
public class Main
    {
        private  Surface m_VideoScreen;
        private  Surface m_FileNameSurface;

      private static  bool m_Terminate = false;

        public Main()
        {
        }

        public void Run()
        {
m_VideoScreen = Video.SetVideoMode(800, 600, 32, false, false, false, true,
true);

Events.Quit += new EventHandler<QuitEventArgs>(Events_Quit);

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

```
Events.UserEvent += new EventHandler<UserEventArgs>(Events_UserEvent);
Thread audioThread = new Thread(new ThreadStart(AudioPlaybackThread));
audioThread.Start();
Events.Run();
        }


        private static void AudioPlaybackThread()
        {
            // Play background Music
            Music bgMusic = new Music("sample4.wav");
            MusicPlayer.Volume = 30;
            MusicPlayer.Load(bgMusic);
            MusicPlayer.Play();

            // Start playing sound Effects
            List<string> audioFiles = new List<string>(new string[] {
            "sample1.ogg", "sample2.ogg", "sample3.ogg" });
            int cnt = 0;
            while (!m_Terminate)
            {
             UserEventArgs userEvent = new
            UserEventArgs(audioFiles[cnt++]);
                if (cnt >= audioFiles.Count)
                    cnt = 0;
                Events.PushUserEvent(userEvent);
                SdlDotNet.Core.Timer.DelaySeconds(2);
            }
        }



        void Events_UserEvent(object sender, UserEventArgs e)
        {
            // Play Ogg File
            if ((e.UserEvent as string).EndsWith("ogg"))
            {
                Sound snd = new Sound( e.UserEvent as string );
                  //m_FileNameSurface = m_Font.Render(( e.UserEvent as
                  string ).Replace( ".
                //ogg", String.Empty ), Color.White );
                snd.Play();
            }
        }

        void Events_Tick(object sender, TickEventArgs e)
        {
            m_VideoScreen.Fill(Color.Black);

            if (m_FileNameSurface != null)
                            m_VideoScreen.Blit(m_FileNameSurface, new
            Point(m_VideoScreen.Width / 2 - m_FileNameSurface.Width / 2,
            m_VideoScreen.Height / 2 - m_FileNameSurface.Height / 2));

            m_VideoScreen.Update();
        }

        void Events_Quit(object sender, QuitEventArgs e)
        {
            m_Terminate = true;
            Events.QuitApplication();
```

```
        }
    }
```

Eerst een korte toelichting over multi-threading mechanisme in .NET

THREADING

Je wil steeds dat je toepassing zo goed mogelijk reageert en alle proces-afhandelingen accuraat en “even belangrijk” uitvoert voor de gebruiker. Dit kan bewerkstelligd worden door multi-threading toe te passen. Threads zijn de basis-eenheden waar het besturingssysteem processor-tijd aan toekent. Besturingssystemen gebruiken processen om de verschillende applicaties die ze uitvoeren te onderscheiden. Het .Net framewerk verdeelt deze processen verder in “Application domains”, waarin 1 of meerdere threads kunnen draaien. Het besturingssysteem wijst processor-tijd toe aan elke thread die uitgevoerd wordt. Deze tijd is kort, zodat het lijkt dat elke thread tegelijkertijd uitgevoerd wordt.

Threads zorgen er dus voor dat het lijkt dat verschillende activiteiten simultaan uitgevoerd worden, bijvoorbeeld: een gebruik wijzigt een pagina, terwijl een andere thread berekeningen aan het uitvoeren is binnen dezelfde toepassing.

Het .Net framewerk voorziet eigen namespaces voor het creëren van threads: System.Threading

Als je doorheen de code gaat, bemerk je enkele nieuwigheden. Ten eerste starten we een nieuwe thread die doorheen een generische lijst van strings loopt en een User Event op de Event Queue zet telkens een sample afgespeeld wordt. Dit is nodig omdat alle SDL operaties op dezelfde thread moeten gebeuren waarop de Event Queue draait (meestal de Main-thread). Audio wordt typisch op een aparte thread afgespeeld.

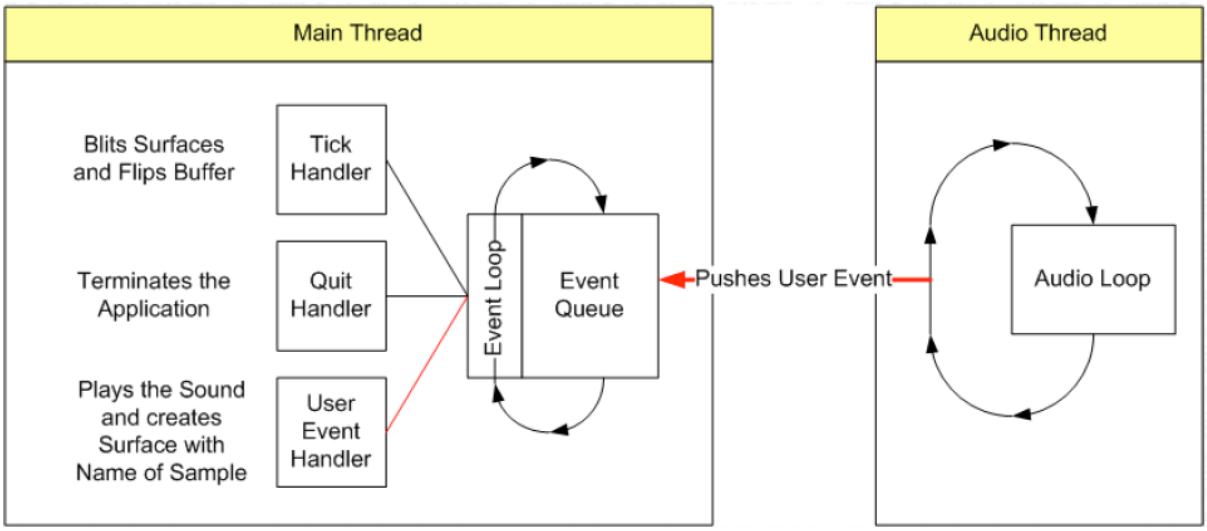
Een thread wordt op deze manier aangemaakt:

```
Thread audioThread = new Thread(new ThreadStart(AudioPlaybackThread));
audioThread.Start();
```

De afhandelingsmethode (AudioPlaybackThread) ziet er zo uit:

```
private static void AudioPlaybackThread()
{
      //je code
```

}



Naast standaard events als Quit event of Tick event bestaat ook het UserEvent, dewelke enkel door de programmeur zelf wordt afgevuurd, en wordt gebruikt om functionaliteit in de Main-Thread uit te voeren.

In de code wordt een bestandsnaam doorgegeven via het UserEventArgs object aan de afhandeling. Deze afhandeling speelt vervolgens de sample af en creëert een Surface met de naam van de sample. Deze surface wordt daarna geblit op het video scherm in het Tick Event.

Verder neemt de Music class een bestand aan dat via de MusicPlayer class geladen is. Om een sample af te spelen zijn volgende handelingen nodig:

1.Laad de sample in een Music instantie door de naam door te geven (of data in a byte[]) aan de constructor.

2.Laad de Music instantie in de MusicPlayer (load methode).

3.Stel de parameters in (zoals volume).

4. Voor de play methode uit.

Bijna hetzelfde voor een kortere sample door de Sound class:

1.Laad de sample in een Sound instantie door de naam door te geven (of data in een byte[]) aan de constructor.

2.Voer de play methode uit

Er zijn verschillende eigenschappen en methodes voor deze classes, bijvoorbeeld door de sample tijd te limiteren door een TimeSpan, of door de sample te faden

Enkele voorbeelden:

```
Channel channel = bang.Play();
channel.SetPanning(0, 255); // Set the panning of the sound on the right.
channel.Pause();  // Pause the sound
channel.Resume(); // Continue the sound
```

Het maximaal aantal kanalen bepaalt het aantal samples die simultaan afgespeeld kunnen worden. Deze staan default op 8. Om het aantal te veranderen gebruik je deze lijn code:

Mixer.ChannelsAllocated = 1000; // zet het maximum aantal op 1000

Telkens de Sample.Play() opgeroepen wordt er een vrij kanaal toegewezen aan de sample vooraleer het afgespeeld wordt. (bijvoorbeeld kanaal 1,2,3 worden samen afgespeeld,en kanaal 4 is vrij, de nieuwe sample krijgt kanaal 4 toegewezen). Let op: als je te weinig kanalen hebt, crasht je programma.

Onderstaand project zal telkens een sample afspelen op een vrij kanaal met een bepaald effect. De linkermuisknop zal de sample links afspelen (het volume van de linkerspeaker), terwijl de rechtermuisknop deze aan de rechterkant afspeelt (volume van de rechterspeaker).

Ontdek zelf volgende functionaliteit: indien je geluidseffecten aan het afspelen bent heb je soms nood aan informatie over een bepaald kanaal:

- Is sound playing
- Is this channel being faded
- Is channel paused
- …

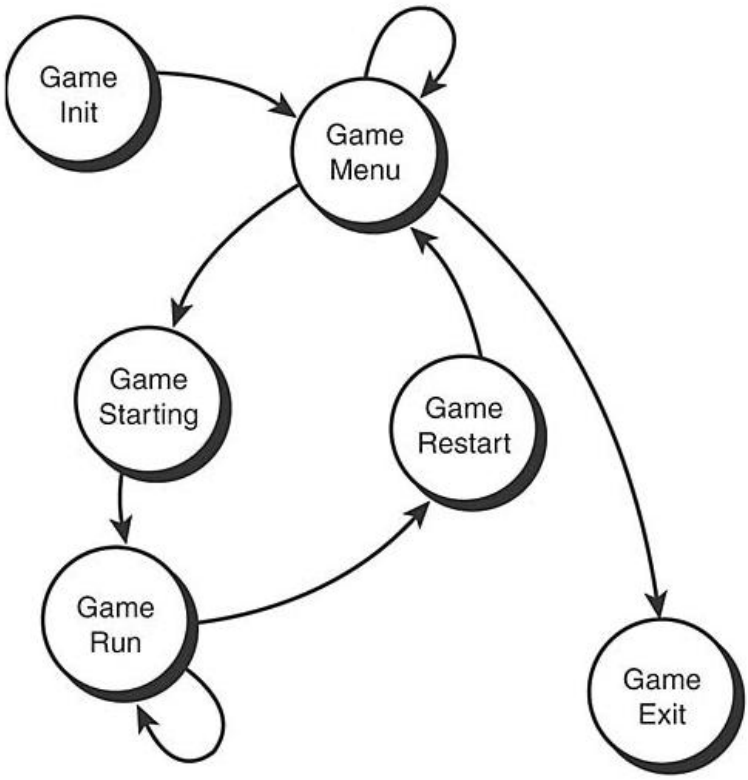
PROJECT 2 – AUDIO EXAMPLE (AUDIOPROJECT1)

```
public class AudioExample
    {
        private const int width = 400;
        private const int height = 100;
        private Surface screen;

        private Sound boing;


        public AudioExample()
        {
            Mixer.ChannelsAllocated = 100;

            Events.MouseButtonDown += new
            EventHandler<SdlDotNet.Input.MouseButtonEventArgs>(Events_Mouse
            ButtonDown);

            Events.Quit += new EventHandler<QuitEventArgs>(Events_Quit);


        }



        void Events_Quit(object sender, QuitEventArgs e)
        {
            Events.QuitApplication();
        }
        Channel c;
      void Events_MouseButtonDown(object sender,
      SdlDotNet.Input.MouseButtonEventArgs e)
        {
            switch (e.Button)
               {
                   case MouseButton.PrimaryButton:
                       // Play on left side
                       boing.Play().SetPanning(205, 50);
```

```
                    break;
                case MouseButton.SecondaryButton:
                    // Play on right side
                    boing.Play().SetPanning(50, 205);

                    break;
            }

        }

        public void Run()
        {
            boing = new Sound("boing.ogg");


            // Start up SDL
            Video.WindowIcon();
            Video.WindowCaption = "SDL.NET - AudioExample";
            screen = Video.SetVideoMode(width, height);

            Events.Run();

        }

    }
```

GAME LOOP

Tijdens de ontwikkeling van je spel, implementeer je best een manager klasse die de verschillende stadiums van je spel bijhoudt. (ook wel een FSM (finite state machine) genoemd).



Deze states worden gedeclareerd via een Enum, een voorbeeld :

```
public enum State
    {
        init,

        game,

        gameover,

        quit

    }
```

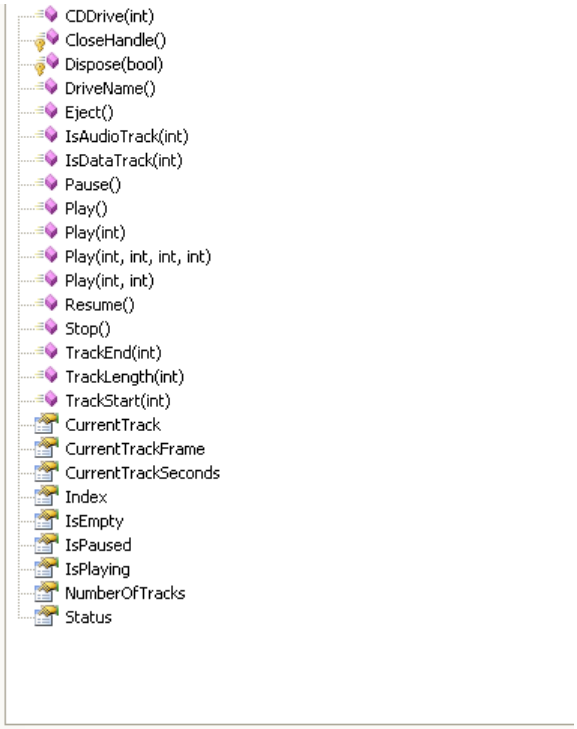
In je TICK event controleer je continu je state :

```
void Events_Tick(object sender, TickEventArgs e)
        {
            //check state
            CheckGameState();
Terwijl de CheckGameState() er bijvoorbeeld als volgt kan uitzien:

private void CheckGameState()
        {

            if (_Manager.TheState == State.init && !_done)
            {
                init();

            }


            else if (_Manager.TheState == State.game && !_done)
            {
                _Background.Fill(Color.Black);

                BuildGame();
                _done= true;


            }

            else if (_Manager.TheState == State.quit) // quit
            {
                Events_Quit(null, null);
            }
ENZ...
```

2.CREER EEN EIGEN CD PLAYER. TOON EEN PLAY, PAUZE, STOP, PREVIOUS EN NEXT KNOP OP HET SCHERM MET BIJHORENDE FUNCTIONALITEIT. TOON OOK HOELANG HET LIED NOG DUURT EN WELK LIEDNUMMER JE AAN HET AFSPELEN BENT.

TIP: SDL.NET heeft een klasse CDDrive met volgende functionaliteit:

```
private CDDrive _drive;

_drive = CDRom.OpenDrive(0);

_drive.Play(_track, _drive.NumberOfTracks - _track);
```



public class **CDDrive** : SdlDotNet.Core.BaseSdlResource
Member of SdlDotNet.Audio

Summary:
Represents an individual CD drive on the system