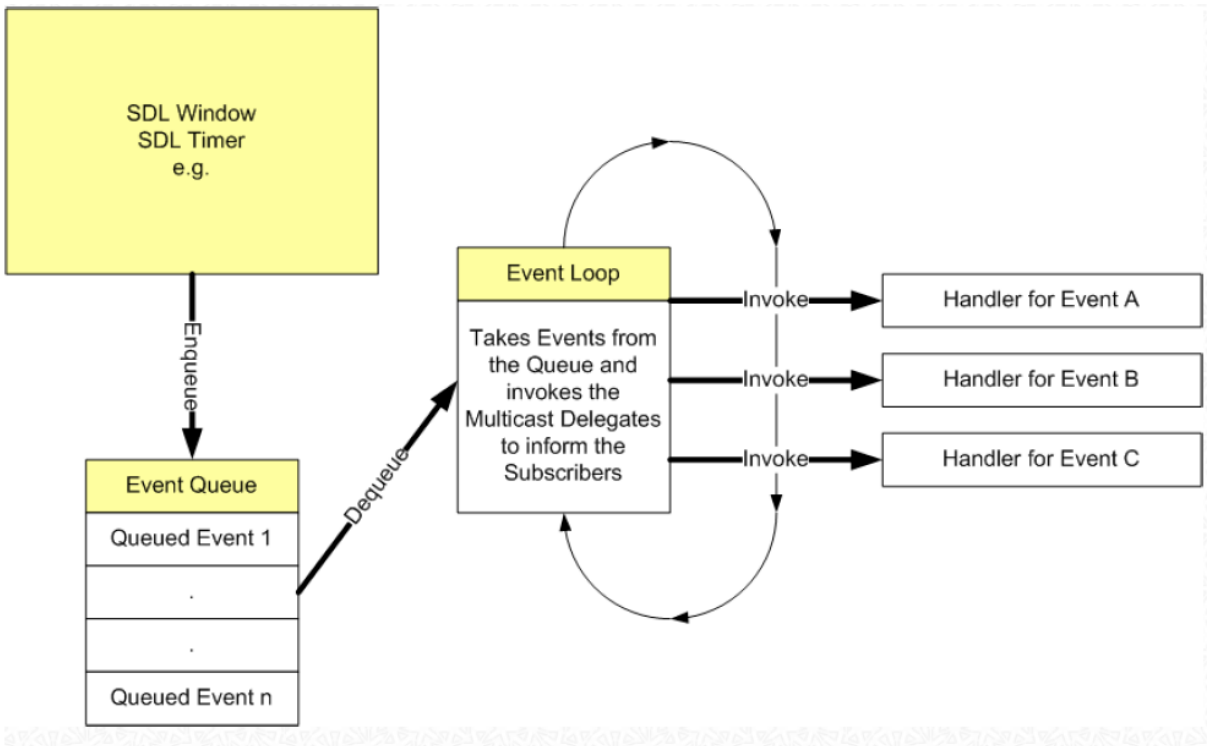
SDL LIBRARY

EVENT DRIVEN PROGRAMMEREN

Een event is simpelweg iets dat gebeurt, zoals een beweging van de muis, een druk op een knop, een vensterbreedte wijziging, …

Als SDL API getriggerd wordt, zal het event in de Event Queue (dat is een data structuur die informatie bijhoudt) worden gezet – ook Dequeuing genoemd. Een Event loop haalt de events van de queue – enqueuing genoemd– met de bijhorende argument data, en start het event, zodat objecten die zich inschrijven deze info te weten komen, en deze verder afhandelen.



Telkens iets gebeurt (muisklik, een timer die afloopt…) wordt een event door de SDL library gegenereerd en bewaard in de event queue. Deze queue wordt ge-de-queued en roept een multicast delegate aan zodat het programma op deze gebeurtenis kan reageren. Daarna gaat men kijken of er een volgend event is.

Events.Run() moet aangeroepen worden om het event sub-systeem te starten.

Een voorbeeld:

Het TICK-event wordt afgevuurd indien zich een “tick” voordoet. Een tick is zoals een timer die periodisch afgaat volgens een bepaalde Frame Rate. (Events heeft een property Fps (zie sessie 1).

Inschrijven op een event gebeurt als volgt:

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

Hier is Events de statische klasse waarop het Tick event wordt getriggerd. Eens een gebeurtenis getriggerd is, zal de afhandeling gebeuren in de Events_Tick functie met als parameter TickEventArgs :

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

Om de Event Loop te starten, wordt in de main **Events.Run()** aangeroepen.

Enkele nuttige events:

AppActive	wanneer programma actief of inactief wordt
ChannelFinished	wanneer een geluidskanaal stopt
KeyboardDown	wanneer een toets van het toetsenbord ingedrukt wordt
KeyBoardUp	wanneer een toets losgelaten wordt
MouseMotion	wanneer met de muis bewogen wordt

Quit	wanneer het programma afgesloten wordt
Tick	elke frame tick wordt dit “tick event” gegenereerd (hartslag van het systeem)
UserEvent	speciaal event – (*zie sessie 5-6)*

PROJECT 1 – KEYBOARD EVENTS

Bijna alle games maken gebruik van keyboard events, om bijvoorbeeld een personage rond te bewegen, of het spel te pauzeren. SDL.Net heeft 2 verschillende events:

- KeyBoardDown
- KeyBoardUp

In onderstaand voorbeeld laden we een achtergrond afbeelding van 800x600 en blitten deze op het Videoscherm met 500x500 px breedte en hoogte. Het indrukken van de pijltjestoetsen zal de gebruiker doorheen de ganse afbeelding scrollen;

```
    enum CursorKeys : int
    {
      Up = 0,
      Down = 1,
      Left = 2,
      Right = 3
    }

  public class KeyBoardTest
  {

      private  Surface m_VideoScreen;
      private  Surface m_BackSurface;
      private  bool[] m_CursorKeys = new bool[4];
      private  int m_BackOffsetX = 0, m_BackOffsetY = 0;

      public KeyBoardTest()
      {

      }

      public void Run()
      {
          m_VideoScreen = Video.SetVideoMode(500, 500, 32, false, false,
          false, true, true);
          LoadImages();

          Events.Quit += new EventHandler<QuitEventArgs>(Events_Quit);
          Events.Tick += new EventHandler<TickEventArgs>(Events_Tick);
          Events.KeyboardDown += new
          EventHandler<SdlDotNet.Input.KeyboardEventArgs>(Events_Keyboard
          );

          Events.KeyboardUp += new
          EventHandler<SdlDotNet.Input.KeyboardEventArgs>(Events_Keyboard
          );

          Events.Run();
      }

      void Events_Keyboard(object sender,
      SdlDotNet.Input.KeyboardEventArgs e)
      {
          switch (e.Key)
          {
              case Key.UpArrow:
                  m_CursorKeys[(int)CursorKeys.Up] = e.Down;
                  break;
              case Key.DownArrow:
                  m_CursorKeys[(int)CursorKeys.Down] = e.Down;
                  break;
              case Key.LeftArrow:
                  m_CursorKeys[(int)CursorKeys.Left] = e.Down;
                  break;
              case Key.RightArrow:
                  m_CursorKeys[(int)CursorKeys.Right] = e.Down;
                  break;
              case Key.Escape:
                  Events.QuitApplication();
                  break;
          }
```

```
        }


        void Events_Tick(object sender, TickEventArgs e)
        {

            List<Point> arrowDestPoints = new List<Point>();
            List<Rectangle> arrowSourceRects = new List<Rectangle>();
            if (m_CursorKeys[(int)CursorKeys.Up])
            {
                if (m_BackOffsetY > 0)
                    m_BackOffsetY -= 5;
            }

            if (m_CursorKeys[(int)CursorKeys.Down])
            {
                if (m_BackOffsetY < m_BackSurface.Height -
m_VideoScreen.Height)
                    m_BackOffsetY += 5;
            }
            if (m_CursorKeys[(int)CursorKeys.Left])
            {
                if (m_BackOffsetX > 0)
                    m_BackOffsetX -= 5;
            }
            if (m_CursorKeys[(int)CursorKeys.Right])
            {
                if (m_BackOffsetX < m_BackSurface.Width -
m_VideoScreen.Width)
                    m_BackOffsetX += 5;
            }

            m_VideoScreen.Blit(m_BackSurface, new Point(0, 0), new
            Rectangle(m_BackOffsetX, m_BackOffsetY, m_VideoScreen.Width,
            m_VideoScreen.Height));

            m_VideoScreen.Update();
        }

        void Events_Quit(object sender, QuitEventArgs e)
        {
            Events.QuitApplication();
        }

        private void LoadImages()
        {
            m_BackSurface = new
            Surface(@"background.jpg").Convert(m_VideoScreen, true, true);
        }

    }
```

Dus we hebben een Surface geïmplementeerd die gescrolld kan worden door de pijltjes toetsen. Wanneer de gebruiker een toets indrukt, wordt de Handler opgeroepen en deze controleert of de ingedrukte toets een pijltjestoets is. Indien deze conditie waar is, wordt een vlag gezet in een bool[4] om de status te bewaren (true = pressed, false=not pressed of

released). Verder wordt in de TICK methode gecontroleerd welke toets ingedrukt is, en wordt de positie van de rechthoek verzet: de X en Y offset voor de achtergrond afbeelding wordt maw bepaald. Tenslotte blitten we deze achtergrond op het video scherm.

De KeyboardEventArgs parameter in de event handler bevat alle informatie om het indrukken van een toets (“keypress”) af te handelen.

```
SdlDotNet.Input.KeyboardEventArgs e)
{
    switch (e.Key)
    {
        case Key.UpArrow:
            m_CursorKeys[(int)CursorKeys.Up] = e.Down;
            break;
    ...
```

Verder heb je misschien bemerkt dat dit event enkel getriggerd wordt wanneer een toets ingedrukt en losgelaten wordt, maar niet wanneer de toets ingedrukt blijft. Dit is omdat de “key-repeating” gedisabled is bij default (Keyboard.EnableKeyReat(delay, rate).

MOUSE EVENTS

Met de muis werken (vanuit oogput van het event mechanisme) is zeer gelijkaardig aan werken met het keyboard. Een muisevent wordt getriggerd onder volgende condities:

-De muis wordt over het Window bewogen (MouseMotion Event)

-Een van de muisknoppen wordt ingedrukt of losgelaten (MouseButtonDown en MouseButtonUp).

– zie oefenreeks -

SPRITE KLASSE

SDL.Net bevat een sprite klasse, die een Surface en Rectangle object bevat. Het Surface object wordt getoond tijdens het blitten op het scherm. De rechthoek/Rectangle wordt gebruikt om een deel van de Surface te tonen.

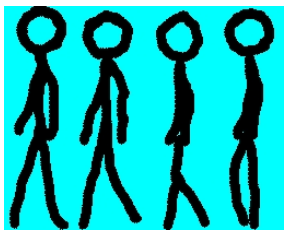
Belangrijk is dat de Sprite klasse een virtuele Update methode bevat, zodat geërfde klasse de mogelijkheid heeft om deze te overschrijven (voor elk event is er een Update methode voorzien).

Ook bevat deze klasse een Render methode die wordt aangeroepen voordat het surface geblit wordt. Meestal bij overerving wordt deze methode gebruikt om nog een taak op het surface te verrichten vooraleer er geblit wordt.

Op zich doet een sprite klasse niet veel, maar door zijn virtuele update methodes is het de bedoeling deze te overschrijven, en kunnen we veel onderhoudbaarder coderen – lees object geörienteerder!

PROJECT 2 : EEN EERSTE ANIMATIE

WE GAAN EEN PERSONAGE VAN LINKS NAAR RECHTS LATEN BEWEGEN OP HET SCHERM. HET BEWEGEN WORDT GECONTROLEERD DOORDAT DE GEBRUIKER OP DE LINKER OF RECHTERPIJL DRUKT.

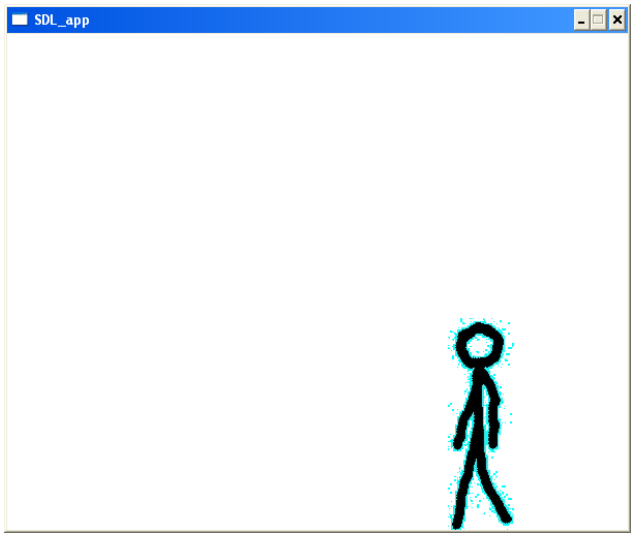


Onze afbeelding bestaat uit 4 sub-afbeeldingen die een “vloeiende loop beweging” voorstelt.

```
enum CursorKeys : int
    {
        Up = 0,
        Down = 1,
        Left = 2,
```

```
        Right = 3
    }

    public class Character : Sprite
    {
        private Surface theVideo;
        private bool[] m_CursorKeys = new bool[4];
        private int offset;
        private int posX;
        private int posY = 0;

        public Character(Surface video)
        {
            theVideo = video;
            posX= theVideo.Width - 64;
            this.Surface = new
            Surface(@"WalkLeft.jpg").Convert(Video.Screen, true, true);
            this.Position = new Point(posX, theVideo.Height - 205);

            this.Surface.Transparent = true;
            this.TransparentColor = Color.FromArgb(0, 255, 255);

        }


        public override void Update(SdlDotNet.Input.KeyboardEventArgs args)
        {
            switch (args.Key)
            {
                case Key.UpArrow:
                    m_CursorKeys[(int)CursorKeys.Up] = args.Down;

                    break;
                case Key.DownArrow:
                    m_CursorKeys[(int)CursorKeys.Down] = args.Down;
                    break;
                case Key.LeftArrow:
                    m_CursorKeys[(int)CursorKeys.Left] = args.Down;

                    break;
                case Key.RightArrow:
                    m_CursorKeys[(int)CursorKeys.Right] = args.Down;
                    break;

            }


        }

        //Draw the character on the video screen
        public override void Update(SdlDotNet.Core.TickEventArgs args)
        {
            //undo previous position first!

            if (offset == 256)
                offset = 0;


            if (m_CursorKeys[(int)CursorKeys.Left])
            {
```

```
                Surface white = new Surface(new Rectangle(new
                Point(offset, 0), new Size(64, 205)));
            white.Fill(Color.White);
                theVideo.Blit(white, new Point(posX, Video.Screen.Height
                - 205));

            posX -= 5;

                theVideo.Blit(this.Surface, new Point(posX,
                Video.Screen.Height - 205), new Rectangle(new
                Point(offset, 0), new Size(64, 205)));

            offset += 64;

        }
        if (m_CursorKeys[(int)CursorKeys.Right])
        {
                Surface white = new Surface(new Rectangle(new
                Point(offset, 0), new Size(64, 205)));
            white.Fill(Color.White);
                theVideo.Blit(white, new Point(posX, Video.Screen.Height
                - 205));

            posX += 5;
                theVideo.Blit(this.Surface, new Point(posX,
                Video.Screen.Height - 205), new Rectangle(new
                Point(offset, 0), new Size(64, 205)));

            offset += 64;

        }
    }
}
```



Merk het volgende op:

PUBLIC CHARACTER(SURFACE VIDEO)

Onze Character klasse is een afgeleide van de Sprite klasse. De Character constructor heeft het Video sub-systeem als parameter. Zoals we weten bestaat een Sprite uit een Surface en een rectangle zodat verder de constructor de Surface een afbeelding geeft en ook de beginpositie waar de afbeelding op het videoscherm moet komen gezet wordt.

PUBLIC OVERRIDE VOID UPDATE(SDLDOTNET.INPUT.KEYBOARDEVENTARGS ARGS)

We overschrijven de virtuele Update-methode van de Sprite klasse voor het Keyboard event en controleren hier welke pijltjestoets ingedrukt is (zie ook Project 1 van deze sessie).

PUBLIC OVERRIDE VOID UPDATE(SDLDOTNET.CORE.TICKEVENTARGS ARGS)

We overschrijven de virtuele Update-methode van de Sprite klasse voor het Tick event en indien de linker pijl ingedrukt gaan we de x positie verschuiven (posX), en hebben we een offset gedefinieerd die telkens een andere sub-afbeelding op het video scherm toont. Onze subafbeelding is 256 pixels breed, dus elke 64 pixels verandert de loop-positie van de figuur (offset+=64). De rechterpijl ingedrukt doet net hetzelfde.

DE MANAGER KLASSE

Om te laten werken heb ik nog een manager klasse geïmplementeerd die het programma als volgt opbouwt:

```
public class Manager
    {
        public Surface m_Video;
        public Manager()
        {
            m_Video = Video.SetVideoMode(600, 480, 32, false, false, false,
            true, true);
            m_Video.Fill(Color.White);
```

```
            Events.Quit += new EventHandler<QuitEventArgs>(Events_Quit);
            Events.Tick += new EventHandler<TickEventArgs>(Events_Tick);
            Events.Fps = 25;
            BuildApp();
            Events.Run();

        }

        void Events_Quit(object sender, QuitEventArgs e)
        {
            Events.QuitApplication();
        }

        void Events_Tick(object sender, TickEventArgs e)
        {
            m_Video.Update();
        }

        private void BuildApp()
        {
            SpriteCollection coll = new SpriteCollection();
            Character ch = new Character(m_Video);

            coll.Add(ch);

            coll.EnableKeyboardEvent();
            coll.EnableKeyboardDownEvent();
            coll.EnableKeyboardUpEvent();
            coll.EnableTickEvent();

        }
    }
```

Belangrijk in deze klasse is de BuidApp() methode. Indien je van sprite objecten gebruikt maakt, moet er een integratie zijn met een SpriteCollection object. Deze SpriteCollection is een generische collectie die sprites verzamelt en hen de events doorgeeft : vb. coll.EnableKeyboardEvent(). Indien deze niet opgeroepen wordt zal je sprite geen events ontvangen! Eens een event getriggerd wordt zal de spritecollectie deze doorgeven aan elke sprite van de lijst.

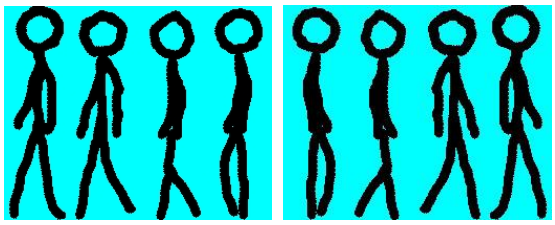
OEFENREEKS

OEFENING 1: TEKEN EEN CIRCEL OP HET SCHERM EN LAAT DEZE ELKE TICK 1 PIXEL IN X EN Y AS OPSCHUIVEN. CONTROLEER WANNEER DE CIRCEL TEGEN DE RAND VAN HET SCHERM BOTST EN LAAT DAN DE CIRCEL NAAR DE ANDERE KANT OPSCHUIVEN. (PRIMITIVES.TXT)

OEFENING 2: BREIDT PROJECT2 – EEN EERSTE ANIMATIE UIT, EN LAAT DE FIGUUR OOK VERTICAAL BEWEGEN. INDIEN DE FIGUUR TEGEN DE GRENZEN VAN HET SCHERM BOTST MAG HIJ NIET VERDER!

OEFENING 3: PAS PROJECT 2 – EEN EERSTE ANIMATIE- AAN ZODAT INDIEN MEN OP DE RECHTERPIJL DRUKT DE LOOP-BEWEGING NAAR RECHTS GERICHT IS.

<< GEBRUIK WALKLEFT.JPG EN WALKRIGHT.JPG >>



OEFENING 4. WE WILLEN EEN BAL 4X PER SECONDE 90° LATEN DRAAIEN NAAR RECHTS TOE.

DE ACHTERGROND VAN DEZE BAL (96X96PX) HEEFT EEN RGB-WAARDE VAN (245,245,245), DAAROM WILLEN WE EERST HET SCHERM MET DEZELFDE ACHTERGRONDKLEUR VAN DE BAL VULLEN ZODAT WE ONZE AFBEELDINGSACHTERGRONDKLEUR VERDOEZELEN.

POSITIONEER OOK DE BAL EXACT IN HET MIDDEN VAN HET SCHERM.

TIP: *BLIT(SURFACE, DESTIONATIONPOSITION, SOURCERECTANGLE)*

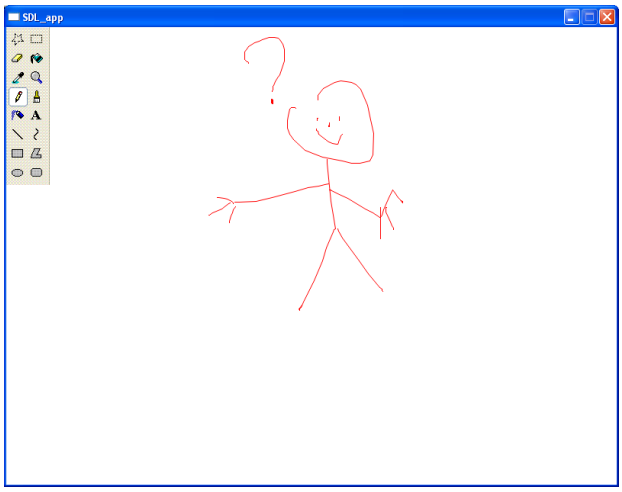
DE ROTATIE GEBEURT MET VOLGENDE CODE:

```
M_BAL        = NEW SURFACE(@"..\..\IMAGES\BAL.GIF");
```

```
SURFACE TMP = NEW SURFACE(M_BAL);

TMP = M_BAL.CREATEROTATEDZOOMEDSURFACE(ROTATION, 1);
```

OEFENING 5:

LATEN WE EEN PAINT NABOUWEN. SIMULEER HET POTLOOD EN TEKEN MET DE MUIS DE BEWEGINGEN NA OP HET SCHERM. OP DE RECHTERMUISKNOP KLIKKEN ZAL ALLES WISSEN.



Deze code krijg je gratis: De MouseMotionHandler is de volgende:

```
m_CursorPosition = e.Position;


if( m_ButtonDown )

{

   IPrimitive line = new SdlDotNet.Graphics.Primitives.Line(
   m_LastCursorPos, m_CursorPosition );

   m_DrawingSurface.Draw( line, Color.Red );

}

m_LastCursorPos = m_CursorPosition;
```

EXTRA:

BREID JE PAINT APPLICATIE UIT. NEEM EEN SCREENSHOT VAN PAINT EN ZORG ER VOOR DAT JE OOK RECHTHOEKEN EN CIRKELS, ELLIPS KAN TEKENEN EN IMPLEMENTEER EEN GOM