# Inhoud

# Inleiding

Verantwoording, totstandkoming idee

Halverwege de 5e klas waren we met natuurkunde bezig met een hoofstuk over harmonische trillingen, waarin onder andere interferentiepatronen behandelt werden.
Aan het eind van dit hoofdstuk beloofde mijn natuurkundeleraar een extra punt voor de eerste die erin slaagde een computerprogramma te schrijven dat in staat was deze interferentiepatronen goed weer te geven. Aangezien ik al een aantal jaar programmeerervaring heb was dit voor mij niet erg moeilijk en had ik in een paar middagen in QuickBASIC een programma voor DOS geschreven dat in staat was in 16 grijstinten interferentie tussen 2 golven weer te geven. Deze weergave zag er ongeveer uit als 2 evengrote steentjes die tegelijkertijd in een bak met water zijn gegooid.
Ik vond het zelf erg leuk om dit te programmeren en verbaasde me over de mooie plaatjes die er met dit programma te maken waren. Daarom ben ik hierna doorgegaan en heb eerst in C een simpele Linux versie van het programma geschreven. Toen ik dat onder de knie had en wist hoe ik het wiskundig voor elkaar kon krijgen om meerdere golven met elkaar te laten interfereren ben ik begonnen aan een C++ library om dit voor een willekeurig aantal sinusgolven met willekeurige parameters te doen.
Bij deze was het eerste begin voor Wave Magic gemaakt. Later bedacht ik me dat ik dit programma voor mijn profielwerkstuk kon gebruiken en ben toen bezig gegaan met het porteren van het library naar Windows (met behulp van Borland C++ Builder). Ook moest er aan het library nog veel werk verricht worden voordat het ook daadwerkelijk zou kunnen werken. Inmiddels is het meeste van dit werk verricht, een grafische interface is ontworpen en het geheel is gelicenseerd onder de GNU Public License.

# Hoofdstuk 1

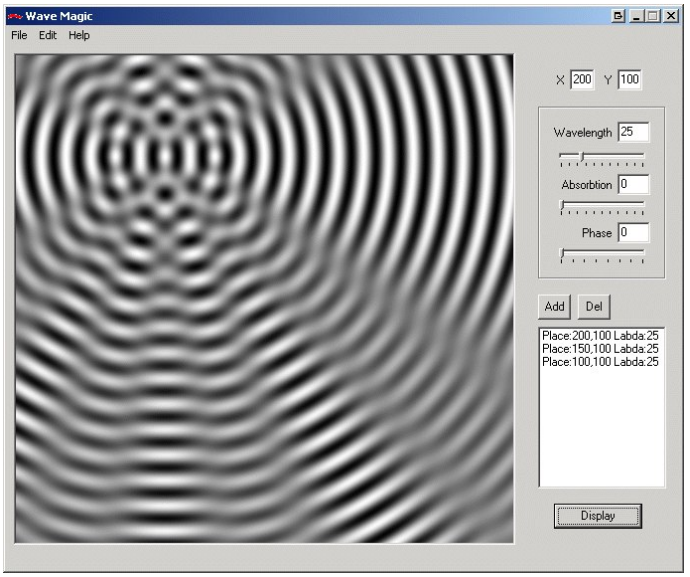
Beschrijving functionaliteit programma

Het programma Wave Magic is eigenlijk met twee doeleinden geschreven.
Ten eerste is er de educatieve functie; het programma is in staat om natuurgetrouw het effect van twee of meer interfererende harmonische trillingen in een plat vlak uit te beelden. Hierdoor is het bijzonder geschikt voor bijvoorbeeld gebruik in de natuurkunde les bij het demonstreren van deze interferentiepatronen zonder een proefopstelling te gebruiken, waardoor dingen als parameters en aantal trillingen gemakkelijker gevarieerd kan worden. Ook zouden leerlingen zelf met het programma kunnen gaan experimenteren om erachter te komen wat voor invloed parameters als labda en afstanden tussen de golven op elkaar hebben.
De andere functie van het programma is gebruik ervan in grafische ontwerpen en andere vormen van ontwerp (bijvoorbeeld websites). Persoonlijk was ik door beelden van het programma (ook in eerdere stadia van ontwikkeling) erg onder de indruk en ben zonder meer overtuigd van de esthetische waarde van de sinusfunctie.

Om allebei deze functies van het programma goed tot hun recht te laten komen was het nodig om met de volgende dingen rekening te houden bij het ontwerpen van zowel de interface naar de achterliggende programmacode als deze achterliggende code zelf:

- De achterliggende wiskunde moest kloppen; de gebruikte formules voor de trillingen, de demping etcetera moeten overeenkomen met een vergelijkbare trilling in de natuur.
- Bij het programma moesten de verschillende parameters zoals demping, labda, fase en de coördinaten makkelijk te veranderen zijn.
- Het moest mogelijk zijn om meerder golfvormen onafhankelijk van elkaar aan een afbeelding toe te voegen of deze er weer uit te verwijderen.
- Voor het ontwerp moest het mogelijk zijn gemakkelijk de beeldinformatie vanuit dit programma naar andere programma's te krijgen.

Uiteindelijk denk ik dat het mij is gelukt, na veel denkwerk, de hierboven genoemde functionaliteit te integreren in het programma. Het is inmiddels mogelijk om een zeer groot aantal golven te tekenen (254) en deze onafhankelijk van elkaar te verwijderen of toe te voegen. Ook is het zeer gemakkelijk om door middel van schuifjes parameters aan te passen, hoewel deze ook als exacte waardes ingevuld kunnen worden. Daarnaast biedt het programma de functionaliteit om simpel afbeeldingen op te slaan als bestanden die in de meeste grafische programma's geopend kunnen worden, plus dat het de mogelijkheid biedt afbeeldingen te kopiëren naar het clipboard van Windows zodat deze in vrijwel elk document geplakt kunnen worden.

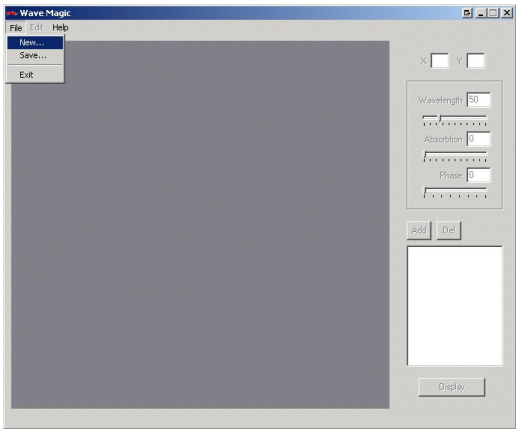


Wave Magic in gebruik

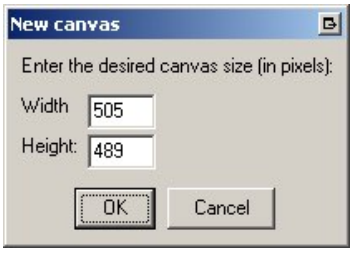
# Hoofdstuk 2

Beschrijving gebruik programma

In dit hoofdstuk zal ik proberen het gebruik van Wave Magic wat verder toe te lichten.
Hiervoor begin ik bij het begin, de installatie. De installatieprocedure van Wave Magic verloopt erg gemakkelijk, het komt erop neer dat het bestand wavemagic-[versie].exe opgestart moet worden (waar x vervangen moet worden door de versie van het programma*). Na de installatieprocedure opgestart te hebben kan er gekozen worden voor een installatiedirectory, dit is waar het uitvoerbare bestand met een aantal bijbehorende dll- bestanden nodig voor de uitvoering van Borland C++ Builder programma's naar toe gekopieerd zal worden. Als vervolgens de daadwerkelijke installatie is afgerond zal er in het menu start een snelkoppeling verschijnen naar Wave Magic.
Als het programma via deze snelkoppeling wordt opgestart verschijnt het hoofdvenster van Wave Magic. Rechts zullen een aantal invulvelden en schuifbalken te zien zijn welke nog niet gebruikt kunnen worden en links daarvan een grijs veld, waar de berekende golfpatronen te zien zullen zijn. Door op File in de menubalk te klikken en vervolgens voor New te kiezen kan een nieuw leeg veld aangemaakt worden waarbinnen golfpatronen berekend kunnen worden. Als hierop wordt geklikt vraagt het programma om een formaat in pixels. Dit is de grootte van het gebied waarbinnen getekend wordt. Deze staat standaard ingesteld op de grootte van het donkergrijze vlak maar kan desgewenst zo groot gemaakt worden als de gebruiker wenst, er zullen dan in het grijze vlak scrollbalken verschijnen.



Het hoofdvenster




Als er een nieuw 'canvas' is gecreëerd worden de invulvelden rechts van het vlak ingeschakeld en zal het eerst grijze veld wit worden en al dan niet scrollbalken krijgen. Op dit moment is het mogelijk sinussen toe te voegen.
Door binnen het witte veld op een willekeurige plek te kiezen kunnen snel en gemakkelijk coördinaten voor de golf ingevoerd worden. Deze kunnen echter ook met de hand ingetoetst worden in de daarvoor bestemde velden X en Y. Vervolgens kan ervoor gekozen worden een andere golflengte bij Wavelength te kiezen. Ook is het mogelijk golven te tekenen met een dempingsfactor in te vullen bij absortion, dit is het aantal pixels waarna een golf volledig uit gedempt is. Als derde is het ook nog mogelijk een fase in te vullen voor de golf bij phase.
Door op 'Add' te klikken wordt de sinusgolf toegevoegd. Ditzelfde kan theoretisch voor maximaal 256 verschillende golven worden herhaald, al zal dit door de hoeveelheid geheugen die dit in beslag neemt dit waarschijnlijk niet haalbaar zijn. Vervolgens kan op 'Display' geklikt worden om alle golven weer te geven binnen het venster. Elke keer dat de weergave vernieuwd moet worden (als er golven zijn toegevoegd of verwijderd) kan op deze knop gedrukt worden om deze wijzigingen toe te passen.
Nadat er op de 'Display' knop is gedrukt kunnen golven desgewenst via 'Edit' gevolgd door 'Copy' in de menubalk naar het Clipboard gekopieerd worden. Ook is het mogelijk via het 'File' menu en dan 'Save...'

Voetnoot:
*De versienummers zijn niet zoals bij de meeste programma's genummerd met een major en een minor nummer (bijvoorbeeld 1.2) maar zijn release candidates (rcx) of het is de final release (genaamd final). Dit heb ik zo gedaan omdat er van dit programma slechts 1 definitieve versie zal zijn, waar in stappen naar toe gewerkt zal worden. Hierna ben ik niet van plan nog verder te ontwikkelen aan (de huidige opzet van) Wave Magic.

het berekende plaatje op te slaan als JPEG of bitmap afbeelding welke in de meeste gangbare grafische programma's geopend kunnen worden. Als er in een later stadium een golf verwijderd moet worden kan deze in het lijstje met golven onder de 'Add' en 'Delete' knoppen geselecteerd worden en door op 'Delete' te klikken kan deze permanent verwijderd worden.
Dit was een korte uitleg van zo ongeveer alle functies van Wave Magic. Aan te raden valt het om vooral eens te experimenteren met golven van verschillende golflengten en aantallen, op deze manier kunnen erg mooie golfpatronen gemaakt worden.

# Hoofdstuk 3

Toelichting gebruikte wiskunde/natuurkunde

De in mijn programma gebruikte natuurkunde is vrij eenvoudig, de daar weer op toegepaste wiskunde is iets complexer maar nog steeds goed te begrijpen.
Voor het tekenen van de harmonische trillingen heb ik de cosinus functie gebruikt, dit om de simpele reden dat een golf op de plek waar hij ontstaat op het moment waarop hij ontstaat altijd de maximale uitwijking heeft.
Vervolgens had ik een functie nodig om voor elke coördinaat (x1,y1) de waarde voor een cosinus op (x2,y2) met gegeven golflengte, fase en demping te berekenen. Hierbij moet ik wel nog vermelden dat over het algemeen op de computer bij grafische applicaties de oorsprong van een assenstelsel op de linkerbovenhoek ligt en dat vanuit die punt naar rechts en beneden voor x en y positieve waarden kunnen worden ingevuld.

Om de waarde van de cosinus te berekenen moet ik allereerst de afstand weten. Deze kan redelijk simpel berekend worden met Pythagoras:

$$d = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$$

Vervolgens kan de afstand a vrij simpel gedeeld worden door de golflengte l. Op deze manier krijg je:

$$u = \cos(d / \lambda)$$

Om vervolgens de fase te corrigeren kan de functie aangepast worden naar:

$$u = \cos((d / \lambda) - \varphi * \lambda)$$

Dan moet er nog dempingsfactor berekend worden. Hiervoor heb ik de volgende functie gebruikt:

$$factor = 1 - (d / demp)^2$$

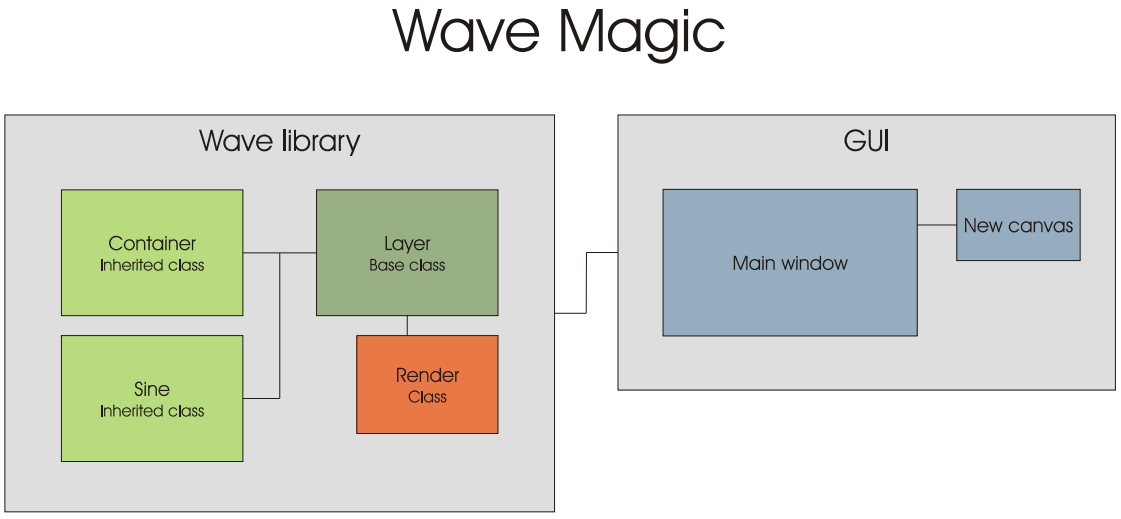
In de programmacode van sine.cpp is te zien hoe ik deze twee functies met elkaar heb gecombineerd zodat voor een compleet matrix aan punten de waarde berekend wordt.

# Hoofdstuk 4

Globale opbouw programmabibliotheek en gebruikte programmeertechnieken

Bij het schrijven van Wave Magic en de achterliggende programmabibliotheek Wavelib heb ik er bewust voor gekozen om hierbij een object-georiënteerde opbouw te gebruiken.
Aan de ene kant heb ik dit gedaan omdat ik graag ervaring op wilde doen met het schrijven in C++ en object-georiënteerd programmeren in het algemeen.
Daarnaast heb ik, door alles object-georiënteerd te maken het voor elkaar gekregen dat vrijwel elk onderdeel van het programma ook los is te gebruiken of is uit te breiden voor een scala aan andere toepassingen. Zoals Wavelib nu in elkaar zit is het bijvoorbeeld zonder al teveel moeite mogelijk om complexe functies zoals fractalen en andere wiskundige algoritmes toe te passen; praktisch elke in getallen uit te drukken functie die in een plat vlak geprojecteerd kan worden is hiervoor te gebruiken.
Ook heb ik ervoor gezorgd dat de grafische gebruikersinterface en de achterliggende code die het echte werk doet vrijwel volledig gescheiden zijn. Hierdoor is het bijvoorbeeld vrij gemakkelijk om een versie van het programma te maken welke zich op een andere manier laat aansturen, bijvoorbeeld door middel van XML-code (dit is een opmaak-taal voor data die in syntax vrijwel gelijk is aan die van het op Internet veel gebruikte HTML). Ook is het op deze manier theoretisch mogelijk Wavelib deel te laten uit maken van een groter geheel, dus ingebouwd in een compleet pakket dat ook veel andere functies kan bevatten.

Ik zal nu de globale structuur van het Wavelib library toelichten, om dit te begrijpen is echter wel een redelijke hoeveelheid kennis van object-georiënteerd programmeren en/of C++ nodig. In de bijbehorende illustratie is ook nog eens simpel te zien welke onderdelen met elkaar samenwerken.
Allereerst heb ik de layer klasse geschreven, welke eigenlijk niets meer of minder is dan een simpel matrix; de klasse bevat contructoren, destructoren en andere methodes voor het aanspreken van een matrix object. De matrix heeft een grootte van (x,y) waaarbinnen voor elke pixel een floating-point waarde wordt onthouden.
Deze layer klasse is de vader van twee andere klassen: de container klasse, de sine klasse. Daarnaast is er de render klasse welke tot doel heeft de matrix van een layer om te zetten in een op het scherm weer te geven afbeelding (bitmap).
De container klasse is een layer klasse die zelf in staat is meerdere layers te bevatten. Het is mogelijk hier layers aan toe te voegen, te verwijderen en te vervangen. Daarnaast kan de inhoud van de layer worden opgevraagd, wat resulteert in het optellen van de waardes van de afzonderlijke layers, gedeelt door het totaal aantal layers.



De sine klasse is de klasse waarin de sinus (of eigenlijk cosinus) functie daadwerkelijk is geïmplementeerd. Hierin wordt onder andere de in het vorige hoofdstuk genoemde wiskunde toegepast. Nadat hierin waarden zijn opgenomen gedraagt deze klasse zich verder als een gewone layer.

Als derde is er de render klasse. Deze berekent voor elke waarde uit het matrix van een layer (welke in de huidige versie van Wavelib tussen de -1 en de 1 dienen te zitten) een integer waarde (opgeslagen in een char) tussen de 0 en de 255 welke op een manier gerangschikt worden waarop de bovenliggende programmalaag (de grafische interface Wave Magic in dit geval) deze kan interpreteren en op het scherm weer kan geven.

De grafische interface van Wave Magic maakt op de volgende manier gebruik van de hierboven uitgelegde principes:

- Als eerste kunnen de parameters voor het aanmaken van een sinus opgegeven worden. Er wordt een container geïnstantieerd.
- Op het moment dat er op Add wordt geklikt wordt er een sinus layer aangemaakt en toegevoegd aan de container.
- Dit zelfde proces kan een aantal keer herhaald worden tot het gewenste aantal sinusgolven bereikt is.
- Hierna kan de gebruiker op Render drukken om de render functie aan te roepen en de teruggegeven data weer te geven binnen het programmavenster.
- Ergens binnen de loop van dit proces kan er een al aangemaakte layer geselecteerd worden en met de Remove functie kan deze layer verwijderd worden uit de container.

# Hoofdstuk 5

Mogelijke uitbreidingen/aanpassingen

Ondanks dat het programma Wave Magic op dit moment officieel af is, zijn er, zoals in elk programma nog wel een aantal dingen die beter zouden kunnen en ook zitten er nog een aantal kleine foutjes in.

Een van de grootste fouten in de opzet van het programma is het feit dat de demping voor meer dan twee golven niet geheel kloppend is. Normaliter is bij een sinusgolf de maximale waarde 1 en de minimale -1. In het geval van een ongedempte sinusgolf kan er dan ook vanuit gegaan worden dat deze waarden voorkomen in een radiale projectie van deze golf.
Het probleem bleek echter te zijn dat als er sprake is van demping in combinatie met meer dan 2 golven het maximum van de som lager is dan de som van het aantal golven. Hierdoor ontstaat, afhankelijk van de hoeveelheid demping en het aantal golven, een vervaging van het signaal.
Omdat de enige oplossing voor dit probleem zou zijn om elke layer te voorzien van een maximum en minimum waarde, welke bij de sinus met demping steeds berekend zou moeten worden was het naar mijn idee niet meer mogelijk deze functionaliteit voor het simpele programma Wave Magic alsnog te implementeren.

Daarnaast was ik al vanaf het begin van plan verschillende soorten rendermechanismen te implementeren, zoals driedimensionale projecties en het gebruik van kleuren om golfpatronen te onderscheiden. Dit was naar mijn idee echter te veel werk om voor mijn profielwerkstuk te doen, misschien dat als het programma ooit nog eens in de praktijk gebruikt gaat worden ik het erin zou schrijven.

In het voorgaande stuk komt het misschien over alsof ik vrij weinig werk heb gehad aan het maken van het programma, echter het tegendeel is waar.
De hierboven genoemde functionaliteit zou me per stuk enkele tientallen uren kosten om in te bouwen, waarmee het totaal aantal uren gemoeid met mijn profielwerkstuk ongeveer op het dubbele zou uitkomen als het aantal uren dat er officieel aan besteed hoort te worden (hoewel ik dat aantal uren nu waarschijnlijk ook al wat overschreden heb).

Zoals ik hierboven vertelde heb ik bepaalde functionaliteit bewust weggelaten als gevolg van tijdgebrek. Hoewel ik deze in de final versie van Wave Magic niet meer zal implementeren, is de kans groot dat ik in de toekomst de afzonderlijke programmabibliotheek Wavelib wel nog zal toepassen. Wavelib is namelijk zo geschreven dat het gemakkelijk aan te spreken is in elk ander programma. Ook heb ik het zo geschreven dat het in principe mogelijk is om het library uit te breiden met andere functies zoals bijvoorbeeld complexe functies (fractalen).
Wave Magic is hierbij eigenlijk niet meer dan een grafische schil om het library heen, ik ben bij het schrijven er grotendeels in geslaagd om front- en backend te scheiden; het eigenlijke werk gebeurd los van de bediening van het programma.

# Hoofdstuk 6

Waarom open source?

Bij het uitbrengen van mijn programma en profielwerkstuk (zowel op internet als het inleveren ervan) heb ik bewust gekozen voor de GNU Public License.
Deze licensie houdt globaal in dat het programma en de bijbehorende documentatie (het profielwerkstuk) alleen gedistribueerd mag worden als de source code erbij beschikbaar is. Daarnaast mag voor het programma, of een afgeleid deel daarvan, geen geld gevraagd worden. Tevens is het andere gebruikers toegestaan vrijelijk gebruik te maken van zowel het programma als de programmacode onder voorwaarde dat hierbij de originele auteur wordt vermeld en dat er geen geld voor wordt gevraagd.
Door het uitbrengen van mijn programma onder deze licensie probeer ik natuurlijk niet in eerste plaats te voorkomen dat het verkocht gaat worden, het gaat mij vooral om de symbolische waarde; naar mijn idee zijn algoritmes niets meer dan wiskunde en ik vind dat niemand wiskundige algoritmes voor zichzelf mag claimen.
Daarnaast mag op deze manier het programma door iedereen gebruikt, aangepast en bekeken worden, wat onder andere voor educatief gebruik van het programma interessant zou zijn.

Zie voor meer informatie de Nederlandse vertaling van de GNU Public License die met dit profielwerkstuk meegeleverd is.

# Afsluiting

Samenvatting en reflectie

Al met al vond ik dit profielwerkstuk erg leuk om te maken. In eerste instantie omdat een groot deel van het werk dat ik ervoor heb gedaan (het programmeren) meer hobby is dan werk. Daarnaast vond ik het leuk om eens de kans te krijgen eens wat meer tijd in één enkele opdracht te steken. Op deze manier vindt er een veel grotere verdieping plaats en daarnaast kan ik niet anders dan toegeven dat ik veel geleerd heb van het maken van mijn profielwerkstuk.
Waarschijnlijk heb ik wel iets meer tijd erin gestoken dan in eerste instantie de bedoeling was maar persoonlijk zit ik hier niet echt mee, het resultaat is, zeker voor een eerste programma in C++, goed gelukt naar mijn idee.

# Appendix A
Broncode van het programma met commentaar

### wavelib.h:
```
#ifndef WAVELIB_H
#define WAVELIB_H

// Pi. But offcourse ;).
#define PI 3.14159265358979323846264338327

// Some default values
#define DEMP_DEF 0
#define LABDA_DEF 25.132741228718345907701147066236

// For later implementation of other render functions;
// B&W, the default render function
#define BW 0

// Dimensions is used to store both coordinates and dimensions of
// layers etc.
struct dimensions {
    unsigned int x, y;
};

#endif
```

### layer.h:
```
#ifndef LAYER_H
#define LAYER_H

#include <string.h>
#include "wavelib.h"

// Base class for all types of layers
class layer {
    protected:
        /* dim contains the horizontal and vertical size
           of the multidimensional array matrix
        */
        dimensions dim;

        /* matrix is a pointer to the 2-dimensional pointer
           array where floating point values for each pixel
           are stored
        */
        float **matrix;

        /* This function is called internally to allocate
           memory for the matrix. It is mainly called by the
           constructor.
        */
        void init(dimensions size);

    public:
        /* The constructor and desctructor, obviously.
           A call to layer is the same as layer(size) where
           size has values of {0,0}.
        */
        layer(dimensions size);
        layer();
        ~layer();
```

```
        /* Code called when, successively, multiplying, dividing,
           adding and equalising two layers.
        */
        layer& operator*(layer &l);
        layer& operator/(layer &l);
        layer& operator+(layer &l);
        void operator=(layer &l);

        /* Idem for floating point numbers
           Note: layer = float will set each pixel value of layer
           to float
           * and / have similar behaviour.
        */
        layer& operator*(float l);
        layer& operator/(float l);
        void operator=(float c);

        /* This might be needed in the future (since = only copies
           the pointers from another layer), as for now it is
           not yet implemented.
        */
        //layer& copy();


        // Get the size of the matrix
        dimensions getsize();

        // Get a pointer to the matrix
        virtual float** getmatrix();
};

#endif
```

**layer.cpp:**

```
#include "layer.h"
#include <windows.h>

layer::layer(dimensions size) {
    init(size);
}

layer::layer() {
    // Init a matrix with default dimensions 0,0
    dimensions b = {0,0};
    init(b);
}

layer::~layer() {
    unsigned int teller;

    for (teller=0;teller<dim.x;teller++) {
        // Delete each row seperately
        delete[] matrix[teller];
    }
}

void layer::init(dimensions size) {
    unsigned int teller;
    dim = size;

    matrix = new float *[dim.x];
```

```
    for (teller=0;teller<dim.x;teller++)
        // Since gcc had some problems compiling new float **[][]
        // I had to do a new statement for every row of pixels
        matrix[teller] = new float[dim.y];
}

dimensions layer::getsize() {
    return dim;
}

float** layer::getmatrix() {
    return matrix;
}

layer& layer::operator*(layer &l) {
    // First check to see if the size of both layers matches
    if ((dim.x != l.getsize().x) || (dim.y != l.getsize().y))
        exit;

    unsigned int x, y;

    // Create a pointer to store the matrix of the layer l
    float **matrix2;
    matrix2 = l.getmatrix();

    // Create an output layer
    layer* me = new layer(getsize());

    for (y=0; y<dim.y; y++) {
        for (x=0; x<dim.x; x++) {
                // Calculate and put the results in me
                me->matrix[x][y] = matrix[x][y]*matrix2[x][y];
        }
    }

    return *me;
}

layer& layer::operator/(layer &l) {
    // First check to see if the size of both layers matches
    if ((dim.x != l.getsize().x) || (dim.y != l.getsize().y))
        exit;

    unsigned int x, y;

    // Create a pointer to store the matrix of the layer l
    float **matrix2;
    matrix2 = l.getmatrix();

    layer* me = new layer(getsize());

    for (y=0; y<dim.y; y++) {
        for (x=0; x<dim.x; x++) {
                // Calculate and put the results in me
                me->matrix[x][y] = matrix[x][y]/matrix2[x][y];
        }
    }

    return *me;
}
```

```
layer& layer::operator+(layer &l) {
    // First check to see if the size of both layers matches
    if ((dim.x != l.getsize().x) || (dim.y != l.getsize().y))
        exit;

    unsigned int x, y;
    layer* me = new layer(getsize());

    for (y=0; y<dim.y; y++) {
        for (x=0; x<dim.x; x++) {
                // Calculate and put the results in me
                me->matrix[x][y] = matrix[x][y]+l.matrix[x][y];
        }
    }

    return *me;
}

layer& layer::operator*(float l) {
    unsigned int x, y;

    layer* me = new layer(getsize());

    for (y=0; y<dim.y; y++) {
        for (x=0; x<dim.x; x++) {
                // Calculate and put the results in me
                me->matrix[x][y] = matrix[x][y]*l;
        }
    }

    return *me;
}

layer& layer::operator/(float l) {
    unsigned int x, y;

    layer* me = new layer(getsize());

    for (y=0; y<dim.y; y++) {
        for (x=0; x<dim.x; x++) {
                // Calculate and put the results in me
                me->matrix[x][y] = matrix[x][y]/l;
        }
    }

    return *me;
}

void layer::operator=(float c) {
    // This function just operates on itself

    unsigned int x, y;

    for (y=0; y<dim.y; y++) {
        for (x=0; x<dim.x; x++) {
                matrix[x][y] = c;
        }
    }
}

void layer::operator=(layer &l) {
        dim = l.getsize();
```

```
        matrix = l.getmatrix();
}
```

**sine.h:**

```
#ifndef SINE_H
#define SINE_H

#include <math.h>

#include "layer.h"

// Basic declaration of sine class, which extends layer with the ability to
render radial sine waves
class sine: public layer {
    public:
        // Default layer constructor
        sine(dimensions size): layer(size) {};

        // Constructor which also calls the draw function
        sine(dimensions size, dimensions place,                              \
 float labda=LABDA_DEF, float demp=DEMP_DEF, float phase=0);

        // Draw a sine wave with the given parameters
        void draw(dimensions place, float labda=LABDA_DEF,                   \
 float demp=DEMP_DEF, float phase=0);
};

#endif
```

**sine.cpp:**

```
#include "sine.h"

sine::sine(dimensions size, dimensions place, float labda,                   \
 float demp, float phase): layer(size) {
    dim = size;
    draw(place, labda, demp, phase);
}

void sine::draw(dimensions place, float labda, float demp, float phase) {
    /* This is where the most complex math (still not too complex
       but well...) of wavemagic takes place.
       In the next few rules one sees how for each pixel a value is
       calculated and put away in the matrix pointer.
    */
    unsigned int teller;

    unsigned int x,y;
    float dst, factor;

    for (x=0;x<dim.x;x++) {
        for (y=0;y<dim.y;y++) {
          /* First, calculate the distance of the current pixel to
             the origin of the wave using Pythagoras and the
             difference between the sine and
             the current pixel.
          */
          dst = hypot(float(place.x)-x, float(place.y)-y);

          /* If we use absorbtion, calculate the absortion factor
             for the current distance to the origin. If none is used
             just set the factor to 1.
```

```
          */
          if (demp) {
              factor=1-sqrt(dst/demp);
              if (factor<0)
                 factor = 0;
          } else {
              factor = 1;
          }

          /* Calculate the value for the current pixel using the
             cosine function and multiply it with factor.
          */
          matrix[x][y] = cos(((dst * 2 * PI)/labda)-phase*labda) * factor;
        }
    }
}
```

**render.h:**

```
#ifndef RENDER_H
#define RENDER_H

#include <math.h>
#include <stdlib.h>

#include "layer.h"

/* The render class takes a layer instance and
   generates a bitmap from it ready for display.
   Currently it takes only a layer with a matrix
   value between -1 and 1, this might be extended
   later on to a larger range.
*/
class render
{
    private:
        // The bitmap where color data will be stored
        unsigned char *buffer;

        // The dimensions of the used layer/bitmap
        dimensions res;

        // A pointer to the matrix with values to be rendered
        float **matrix;

        // Render initialisation function, does memory allocation and
        // related stuff
        void init(dimensions size);

    public:
        // Render constructor from a layer object
        render(layer* l);

        // Render contructor from a layer object
        // Use dimensions to crop in the layer object
        render(layer* l, dimensions size);

        // Normal constructor, creates an empty render (not in use yet!)
        render();

        // Destructor, frees memory
        ~render();
```

```
        // Returns the bitmap buffer
        unsigned char* getbitmap();

        // Returns the dimensions in pixels (res)
        dimensions getsize();

        // Perform the render proces
        unsigned char* run(int method=BW);
};

float rintf(float x);
#endif
```

**render.cpp:**

```
#include "render.h"
/* Since no color support is built in (yet) we might
   as well disable it here to save some memory and gain speed.
*/
#define NUMCOLORS 1

unsigned char* render::getbitmap() {
    // Return pointer to the internal bitmap representation
    return buffer;
}

void render::init(dimensions size) {
    // Make the given resolution our own resolution
    res = size;

    //  Allocate memory for the bitmap
    buffer = new unsigned char[NUMCOLORS*res.x*res.y];
}

render::~render() {
    // Whoppa, clean out the buffer :P
    delete[] buffer;
}
render::render(layer* l) {
    // Initialise from layer
    init(l->getsize());

    // Give ourselves access to the matrix of the layer
    matrix = l->getmatrix();
}

render::render(layer* l, dimensions size) {
    /* Initialise our bitmap using the custom size
       (watch out, there's ABSOLUTELY no check for the size,
       this should be checked from a higher level in the program
       or maybe implemented in the future)
    */
    init(size);
    matrix = l->getmatrix();
}

unsigned char* render::run(int method) {
    // There rendering, this is where it all happens =)

    unsigned int teller=0,teller1,x,y;
    unsigned char t;

    // For each pixel calculate the color value between 0 and 254
```

```
    for (y=0;y<res.y;y++) {
        for (x=0;x<res.x;x++) {
                /* Since we're setting NUMCOLORS times the same
                   value I've chosen to just use a single variable
                   for now.

                   Since our input (matrix) is between -1 and 1 we
                   can simply multiply it with 127 (254/2) and add
                   128 to it to make it positive.
                */
                t = (unsigned char)rintf(matrix[x][y] * 127) + 128;

                /* For each color pane (3 for RGB colors, 4 for
                   RGBA, 1 for grayscale, we need to add the next
                   value to buffer.
                  */
                for (teller1=0;teller1<NUMCOLORS;teller1++) {
                 *(buffer+teller) = t;
                        teller++;
                }
        }
    }

    /* Return the pointer to the buffer so we don't HAVE to use
       getbitmap() from the calling function.
    */
    return getbitmap();
}

dimensions render::getsize() {
    /* Do I have to explain this one? :P
       For more information about the struct getsize see
       wavelib.h
    */
    return res;
}

float rintf(float x) {
        /* I had to manually implement this function in C++ Builder
           because well... it just doesn't come with the package.
           (as it does in Linux)
        */
        if (fmod(x,1) > 0.5) {
                return (float)ceil(x);
        } else {
                return (float)floor(x);
        }

}
```

**container.h:**

```
#ifndef CONTAINER_H
#define CONTAINER_H

#include "layer.h"

/* The container classe is a class which holds a list of
   multiple layer instances. With the use of methods it's possible
   to add, delete and replace different layers.
   But now comes the magic: it is able to automatically sum up the
   layers, returning one layer with values between -1 and 1. In this
```

```
   behaveour it acts just like a normal layer.
*/
class container: public layer {
    protected:
        // Current number of layers in use
        int curr;

        /* This isn't very elegant, limits the number of layer to
           255. On the other hand, who the f*ck would need more than
           254 layers!?
           BTW this is the pointer array which contains the layer
           stack.
        */
        layer* layers[255];
    public:
        /* The constructor overhere inherits from layer, plus
           that it initialises curr to 0.
        */
        container(dimensions size): layer(size) { curr=0; };

        // Destructor well.. just deallocates memory.
        ~container();

        // Two functions for layer interoperability
        // For more comments check the layer class
        void operator=(layer &l);
        void operator=(float c);
        float** getmatrix();

        // At a layer after the last one
        int append(layer* l);
        /* Insert a layer at pos, making the other layers shift
           in the stack
        */
        int insert(layer* l, int pos);
        /* Remove layer at pos, making the other layers on the
           stack close up
        */
        int remove(int pos);
        // Replace layer at pos with l.
        int replace(layer* l, int pos);
};

#endif
```

**container.cpp:**

```
#include "container.h"
container::~container() {
    int teller;
    for (teller=0;teller<curr;teller++) {
        // Call the destructor for each individual layer
        delete layers[teller];
    }
}

int container::append(layer* l) {
    // Set the next layer to contain l and increase curr
    // with 1 while on the job.
    layers[curr++] = l;
    return 0;
}
```

```
int container::replace(layer* l, int pos) {
    if (curr>pos) {
        //Replace layer pos with l
        layers[pos] = l;
        return 0;
    } else {
        // Error, the layer can't be added because
        // the total number  layers is too low.
        return -1;
    }
}

int container::insert(layer* l, int pos) {
    int teller;

    // First let all the layers in the stack make room for
    // the new layer.
    for (teller=curr; teller>=pos; teller--)
        layers[teller+1] = layers[teller];

    // Let the new layer move to it's position.
    layers[pos] = l;

    // Increase the layer counter with 1.
    curr++;

    // Let the caller know everythin went ok.
    return 0;
}

int container::remove(int pos) {
    // Check to see if there is indeed a layer available for
    // deletion
    if (pos>=curr) return -1;

    // Call the destructor for the layer
    delete layers[pos];

    // Make all the other layers close up.
    int teller;
    for (teller=pos; teller<curr; teller++)
        layers[teller] = layers[teller+1];

    // Update the layer counter
    curr--;

    // Let them callers know we did it right
    return 0;
}

void container::operator=(float c) {
    // Check the layer class for the corresponding function
    // for details.
    unsigned int x, y;

    for (y=0; y<dim.y; y++) {
        for (x=0; x<dim.x; x++) {
                matrix[x][y] = c;
        }
    }
}
```

```
void container::operator=(layer &l) {
    // Check the layer class for the corresponding function
    // for details.
    dim = l.getsize();
    matrix = l.getmatrix();
}

float** container::getmatrix() {
    /* This is where the real party takes place :P
       Overhere we sum up the layers one by one while dividing them
       by the total number of layers.
    */
    int teller;

    if (curr>0) {
        // Do the first layer manually since we can't add it up to
        // itself
        *this = (*layers[0])/curr;

        for (teller=1; teller<curr; teller++)
            *this = *this + (*layers[teller])/curr;

    } else {
        // If there are no layers set, just return a layer with 0
        // values.
        *this = 0.0;

    }

    // Return our matrix as we should, being a nice layer
    return matrix;
}
```

**main.h:**

```
//---------------------------------------------------------------------
#ifndef mainH
#define mainH
//---------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <StrUtils.hpp>
#include "sine.h"
#include "render.h"
#include "layer.h"
#include "container.h"
#include "new.h"
#include <ExtCtrls.hpp>
#include <jpeg.hpp>
#include <Menus.hpp>
#include <ComCtrls.hpp>
#include <Dialogs.hpp>
#include <ExtDlgs.hpp>
#include <Clipbrd.hpp>
//---------------------------------------------------------------------
class TForm1 : public TForm
{
__published:  // IDE-managed Components
        TButton *Button1;
        TScrollBox *ScrollBox1;
        TMainMenu *MainMenu1;
```

```
        TMenuItem *File1;
        TMenuItem *Exit1;
        TMenuItem *N1;
        TMenuItem *Save1;
        TMenuItem *New1;
        TLabeledEdit *xpos;
        TLabeledEdit *ypos;
        TLabeledEdit *labda;
        TLabeledEdit *phi;
        TLabeledEdit *demp;
        TTrackBar *TrackBar1;
        TTrackBar *TrackBar2;
        TTrackBar *TrackBar3;
        TGroupBox *GroupBox1;
        TSavePictureDialog *SavePictureDialog1;
        TMenuItem *Help1;
        TMenuItem *About1;
        TMenuItem *Edit1;
        TMenuItem *Copy1;
        TListBox *List;
        TButton *Add;
        TButton *Del;
        TPanel *Panel1;
        TImage *Image1;
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall Exit1Click(TObject *Sender);
        void __fastcall New1Click(TObject *Sender);
        void __fastcall Image1MouseUp(TObject *Sender,                    \
 TMouseButton Button, TShiftState Shift, int X, int Y);
        void __fastcall TrackBar2Change(TObject *Sender);
        void __fastcall TrackBar1Change(TObject *Sender);
        void __fastcall TrackBar3Change(TObject *Sender);
        void __fastcall Save1Click(TObject *Sender);
        void __fastcall About1Click(TObject *Sender);
        void __fastcall Copy1Click(TObject *Sender);
        void __fastcall AddClick(TObject *Sender);
        void __fastcall DelClick(TObject *Sender);
private: // User declarations
public:       // User declarations
        __fastcall TForm1(TComponent* Owner);
        void initCanvas(dimensions size);
};
//---------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------
#endif
```

**main.cpp:**

```
//---------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "main.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"

container* laag;

float teller=0;
```

```
TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
void __fastcall TForm1::Button1Click(TObject *Sender)
{
        render rnd = laag;
        unsigned char* buffer = rnd.run();
        int teller = 0;
        unsigned char* ptr;

        for (int y = 0; y < Image1->Picture->Bitmap->Height; y++) {
                ptr = (unsigned char*)Image1->Picture->Bitmap->ScanLine[y];
                for (int x = 0; x < Image1->Picture->Bitmap->Width; x++) {
                        ptr[x*3] = buffer[x+teller];
                        ptr[x*3+1] = buffer[x+teller];
                        ptr[x*3+2] = buffer[x+teller];
                }
                teller += Image1->Picture->Bitmap->Width;
        }
        Image1->Refresh();
}
//---------------------------------------------------------------------

void __fastcall TForm1::Exit1Click(TObject *Sender)
{
        exit(EXIT_SUCCESS);
}
//---------------------------------------------------------------------

void __fastcall TForm1::New1Click(TObject *Sender)
{
        Form2->Visible = True;
}
//---------------------------------------------------------------------

void TForm1::initCanvas(dimensions size) {
        delete laag;
        laag = new container(size);
        if (Image1->Picture->Bitmap->HandleAllocated()) {
                Image1->Picture->Bitmap->ReleaseHandle();
                DeleteObject(Image1->Picture->Bitmap->Handle);
        }

        // Initialize UI
        List->Clear();
        Image1->Visible = True;
        Edit1->Enabled = True;
        labda->Enabled = True;
        phi->Enabled = True;
        demp->Enabled = True;
        TrackBar1->Enabled = True;
        TrackBar2->Enabled = True;
        TrackBar3->Enabled = True;
        xpos->Enabled = True;
        ypos->Enabled = True;
        List->Enabled = True;
        Add->Enabled = True;
        Del->Enabled = True;
        Button1->Enabled = True;
```

```
        // Initialise our bitmap
        // (in a later version, maybe, a new bitmap should be created)
        Image1->Picture->Bitmap->PixelFormat = pf24bit;
        Image1->Picture->Bitmap->Width = size.x;
        Image1->Picture->Bitmap->Height = size.y;
}

void __fastcall TForm1::Image1MouseUp(TObject *Sender,                    \
 TMouseButton Button, TShiftState Shift, int X, int Y)
{
        xpos->Text = IntToStr(X);
        ypos->Text = IntToStr(Y);
}

void __fastcall TForm1::TrackBar2Change(TObject *Sender)
{
        labda->Text = TrackBar2->Position;
}
//---------------------------------------------------------------------

void __fastcall TForm1::TrackBar1Change(TObject *Sender)
{
        demp->Text = TrackBar1->Position;
}
//---------------------------------------------------------------------

void __fastcall TForm1::TrackBar3Change(TObject *Sender)
{
        phi->Text = TrackBar3->Position/31.4;
}
//---------------------------------------------------------------------

void __fastcall TForm1::Save1Click(TObject *Sender)
{
        if (!Image1->Picture->Bitmap->HandleAllocated()) {
                MessageDlg("No canvas found to save!\nPlease              \
 create a canvas first.", mtError, TMsgDlgButtons() << mbOK, 0);
                return;
        }

        if (SavePictureDialog1->Execute()) {
                AnsiString fn;
                fn = SavePictureDialog1->FileName;

                if (AnsiEndsStr(".bmp", fn)) {
                        Image1->Picture->SaveToFile                       \
(SavePictureDialog1->FileName);
                } else {
                        TJPEGImage *jp = new TJPEGImage();
                        jp->Assign(Image1->Picture->Bitmap);
                        jp->SaveToFile(SavePictureDialog1->FileName);
                        delete jp;
                }
        }
}
//---------------------------------------------------------------------


void __fastcall TForm1::About1Click(TObject *Sender)
{
        MessageDlg("Wave Magic is written by Mathijs de Bruin             \
```

```
 <drbob@dokterbob.net>.\n\nCopyright (C) 2002-2003 Free                    \
 Software Foundation, Inc.\nThis program is free software and              \
 may be distributed according to the terms of the GNU General              \
 Public License.\n", mtInformation, TMsgDlgButtons() << mbYesToAll, 0);
}
//---------------------------------------------------------------------

void __fastcall TForm1::Copy1Click(TObject *Sender)
{
        Clipboard()->Assign(Image1->Picture);
}
//---------------------------------------------------------------------

void __fastcall TForm1::AddClick(TObject *Sender)
{
        int level;
        dimensions plaats = { StrToInt(xpos->Text), StrToInt(ypos->Text)};

        level = laag->append(new sine(laag->getsize(), plaats,             \
 StrToFloat(labda->Text), StrToFloat(demp->Text), StrToFloat(phi->Text) ));
        List->Items->Insert(level, "Place:"+xpos->Text+",                  \
 "+ypos->Text+" Labda:"+labda->Text);
}
//---------------------------------------------------------------------

void __fastcall TForm1::DelClick(TObject *Sender)
{
        int level = List->ItemIndex;
        laag->remove(level);
        List->Items->Delete(level);
}
//---------------------------------------------------------------------
```

**new.h:**

```
//---------------------------------------------------------------------
#ifndef newH
#define newH
//---------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include "main.h"
//---------------------------------------------------------------------
class TForm2 : public TForm
{
__published:  // IDE-managed Components
        TLabel *Label1;
        TEdit *width;
        TEdit *height;
        TButton *Button1;
        TButton *Button2;
        TStaticText *StaticText1;
        TStaticText *StaticText2;
        void __fastcall Button2Click(TObject *Sender);
        void __fastcall Button1Click(TObject *Sender);
private:  // User declarations
public:       // User declarations
        __fastcall TForm2(TComponent* Owner);
};
//---------------------------------------------------------------------
```

```
extern PACKAGE TForm2 *Form2;
//---------------------------------------------------------------------------
#endif
```

**new.cpp:**

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "new.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm2 *Form2;
__fastcall TForm2::TForm2(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Button2Click(TObject *Sender)
{
        Form2->Visible = False;
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Button1Click(TObject *Sender)
{
        Form2->Visible = False;
        dimensions size = {     StrToInt(width->Text),
                                StrToInt(height->Text)};
        Form1->initCanvas(size);
}
//---------------------------------------------------------------------------
```

# Appendix B
GNU Public License

Dit is een niet officiële vertaling van de GNU Algemene Publieke Licentie in het Nederlands. Deze licentie is niet gepubliceerd door de Free Software Foundation, de condities van software onder de GPL hieronder zijn niet rechtsgeldig. Enkel de originele Engelse tekst van de GNU GPL bevat geldige richtlijnen. Daarentegen hopen we dat deze vertaling de Nederlandstaligen helpt om de GNU GPL beter te begrijpen.



Voorwoord

De licenties van de meeste software zijn zo opgesteld om U het recht te ontnemen om die software te delen en te wijzigen. Hier tegenover staat de GNU Algemene Publieke Licentie, die bedoeld is om U de vrijheid te garanderen dat U de software kan delen en wijzigen -- om er zeker van te zijn dat de software vrij is voor alle gebruikers. Deze Algemene Publieke Licentie is van toepassing op het merendeel van de Free Software Foundation's software en van alle andere programma's waarvan de auteur ze plaatst onder deze licentie.
(Sommige software van de Free Software Foundation is gedekt door de GNU Algemene in de Publieke Licentie). U kan deze ook toepassen op uw eigen programma's.

Wanneer we het hebben over vrije software, dan hebben we het over vrijheid, niet prijs. Onze Algemene Publieke Licentie laat u toe om kopieën te verspreiden van vrije software (en dat U geld kan vragen voor deze dienst) en dat U er de broncode van hebt of kan krijgen als U dat wenst, dat U de software kan wijzigen of er delen van kan gebruiken in nieuwe vrije programma's en dat U weet dat U deze dingen kan doen.

Om deze rechten te beschermen, moeten we verbieden dat iemand U deze rechten ontzegt of vraagt deze op te geven. Deze restricties brengen enkele verantwoordelijkheden mee indien U kopieÙn van de software verspreidt of de software wijzigt.

Bijvoorbeeld, als U kopieën van zulk programma verspreidt, kostenloos of voor een vergoeding, dan moet U de personen die de software ontvangen al de rechten geven die U hebt. U moet uzelf ervan verzekeren dan ook zij de broncode ontvangen of kunnen verkrijgen. U moet hen ook deze licentie tonen zodat ze hun rechten kennen.

We beschermen uw rechten met twee stappen
(1) de software wordt auteursrechtelijk beschermd, en
(2) we bieden U deze licentie die U de legale toestemming geeft om de software te kopiëren, te verspreiden en/of te wijzigen.

Alsook willen we voor de bescherming van de auteur en onszelf iedereen ervan verzekeren dat er geen garantie is voor deze vrije software. Als de software gewijzigd is door iemand anders en doorgegeven, dan willen we dat de ontvanger weet dat wat ze ontvangen hebben niet het origineel is, zodat problemen veroorzaakt door anderen geen effect hebben op de reputatie van de oorspronkelijke auteur.

Ten laatste, elk vrij programma wordt voortdurend bedreigd door software patenten. We wensen het gevaar te vermijden dat de verdelers van een vrij programma uiteindelijk een patent verkrijgen op het programma en het daarmee in eigendom van een particulier brengen. Om dit te vermijden, hebben we het duidelijk gemaakt dat elk patent in licentie gegeven moet zijn voor eenieders vrij gebruik, oftewel helemaal niet in licentie gegeven mag zijn.

De exacte bepalingen en condities om te kopiëren, verspreiden en wijzigen volgen hieronder.

GNU ALGEMENE PUBLIEKE LICENTIE
BEPALINGEN EN VOORWAARDEN OM TE KOPIËREN, VERSPREIDEN EN WIJZIGEN

0. Deze licentie is van toepassing op elk programma of ander werk dat een notie bevat van de eigenaar die zegt dat het verspreid mag worden onder de bepalingen van deze licentie. Het "Programma", verder in de tekst, verwijst naar eender zulk programma of werk, en een "werk gebaseerd op het programma" verwijst naar het Programma of eender welk ander afgeleid werk onder de wet van het auteursrecht: dit wil zeggen, een werk dat het Programma of een deel ervan bevat, letterlijk oftewel gewijzigd en/of vertaald naar een andere taal.
(Hierna vallen vertalingen zonder beperking onder de term "wijziging".)
Elke licentiehouder wordt geadresseerd als "u".

Andere handelingen dan kopiëren, verspreiden en wijzigen zijn niet gedekt door deze licentie; hiervoor is deze licentie niet bedoeld. De handeling om het Programma uit te voeren is niet gelimiteerd, en de uitvoer van het Programma is enkel gedekt als de inhoud bestaat uit een werk gebaseerd op het Programma (onafhankelijk of deze uitvoer gemaakt is door het Programma uit te voeren). Of dit waar is hangt af van wat het Programma doet.

1. U mag letterlijke exemplaren verspreiden van de programma broncode en deze kopiëren zoals U deze ontvangt, in eender welke vorm, op voorwaarde dat U ervoor oplet dat U op elke kopie de gepaste auteursrechten en afwijzing van garantie vermeldt; hou alle referenties naar deze licentie en naar het ontbreken van garantie intact ;en geef aan elke andere ontvanger van het Programma een kopie van deze licentie, bijgevoegd bij het Programma.

U mag een honorarium vragen voor de fysische daad van het afleveren van een kopie, en U mag indien U dat wenst een garantie bescherming bieden voor een honorarium.

2. U mag uw kopie of kopijen van het Programma, of een deel van het Programma, wijzigen, daarbij een werk gebaseerd op het Programma vormend. U mag deze wijzigingen kopiëren en verspreiden onder de bepalingen van Paragraaf 1 hierboven, indien U ook aan al deze voorwaarden voldoet:

a) U moet in de gewijzigde bestanden duidelijk vermelden dat U het bestand gewijzigd hebt en de datum waarop U dat gedaan hebt.

b) U moet elk werk dat U publiceert of verspreidt en dat volledig of gedeeltelijk bestaat uit het Programma, of daarvan een afgeleid werk is, als een geheel in licentie geven, zonder kosten, aan alle derde partijen onder de bepalingen van deze Licentie.

c) Indien het gewijzigde Programma normaal gezien interactief parameters inleest, dan moet U er voor zorgen dat wanneer het Programma zonder deze parameters gestart wordt, het een boodschap weergeeft met een gepast auteursrechtbericht en een mededeling dat er geen garantie is (of anders, dat U een garantie voorziet) en dat gebruikers het Programma mogen verspreiden onder deze voorwaarden. De boodschap moet de gebruiker ook duidelijk maken hoe hij een kopij van deze Licentie kan bekijken.

(Uitzondering : als het Programma zelf interactief is en normaal geen boodschap toont, dan is het niet vereist dat uw werk gebaseerd op dit Programma zulk een boodschap weergeeft.

Deze vereisten zijn van toepassing op het werk als een geheel. Als herkenbare stukken van dat werk niet afgeleid zijn van het Programma, en redelijkerwijs onafhankelijk beschouwd kunnen worden, dan is deze licentie, en zijn bepalingen, niet van toepassing op die delen als U die als aparte werken verspreidt. Maar als U die zelfde delen verspreidt als deel van een geheel dat een werk is gebaseerd op het Programma, dan moet de verspreiding van het geheel op de bepalingen van deze licentie geschieden, dewelke's vergunningen voor andere licentiehouders zich uitbreiden tot het volledige geheel, en dus tot elke deel van het werk, onafhankelijk van wie het geschreven heeft.

Dus, het is niet de bedoeling van deze sectie om uw rechten op te eisen of te wedijveren om uw rechten op werk dat geheel door uzelf geschreven is, het is eerder de bedoeling het recht controle uit te oefenen mogelijk te maken op de verspreiding van afgeleide of collectieve werken gebaseerd op het Programma.

Daarenboven, de bundeling van een werk niet gebaseerd op het Programma met het Programma (of met een werk gebaseerd op het Programma) op een opslagmedium of verspreidingsmedium brengt het ander werk niet onder deze licentie.

3. U mag het Programma, of een werk gebaseerd op het Programma, zie paragraaf 2, verspreiden en kopiëren, in binaire of uitvoerbare vorm onder de bepalingen van paragraaf 1 en 2 hierboven, op voorwaarde dat U aan een van de volgende voorwaarden voldoet :

a) Voeg een volledige overeenkomende broncode bij, leesbaar door computers, verspreid onder de bepalingen van de paragrafen 1 en 2, op een medium dat gebruikelijk is voor het uitwisselen van software; of,

b) Voeg een voor minstens 3 jaar geldige, geschreven, offerte bij, om de complete overeenstemmende broncode, op een medium dat hiervoor gebruikelijk is, voor Computers leesbaar, verspreidbaar onder de bepalingen van de paragrafen 1 en 2 hierboven, aan elke derde partij te leveren, voor een vergoeding die niet meer bedraagt dan de kost om de broncode te kopiëren.

c) Voeg de informatie bij die U ontving betreffende het aanbod om de bijpassende broncode te verkrijgen. (Dit alternatief is enkel toegestaan voor niet commerciële verspreiding en enkel als U het programma in binaire of uitvoerbare vorm ontving met zulk een aanbod, in overeenstemming met subparagraaf b erboven.) De broncode van een werk is de vorm van het werk waaraan voorkeur wordt gegeven om er wijzigingen in aan te brengen. Voor een uitvoerbaar werk betekent volledige broncode alle code van alle modules waar het werk uit bestaat, en daarbovenop alle definitie bestanden van de interface(s) en alle scripts om het programma te compileren en het uitvoerbare bestand te installeren. Als een speciale uitzondering moet de verspreide broncode niets bevatten dat normaal verspreid (in broncode of uitvoerbare vorm ) wordt met de hoofdcomponenten (compiler, kernel, enz...) van het besturingssysteem op welke het Programma draait, tenzij die component bij het uitvoerbare bestand zit.

Als verspreiding van een uitvoerbaar bestand of binaire code mogelijk gemaakt wordt door toegang tot het kopiëren van een vooraf bepaalde plaats, dan telt het mogelijk maken de broncode van diezelfde plaats te kopiëren als het verspreiden van de broncode, zelfs indien het mee kopiëren van de broncode optioneel is.

4. U mag het Programma niet kopiëren, wijzigen, verder in licentie geven of verspreiden behalve zoals expliciet vermeld in deze licentie. Eender welke poging om het programma op een andere manier te

kopiëren, wijzigen, verder in licentie geven of verspreiden is ongeldig en verklaart automatisch uw rechten bepaald in deze licentie nietig. Derde partijen die kopieën of rechten van U hebben ontvangen onder deze licentie blijven hun rechten behouden zolang ze de voorwaarden niet schenden.

5. U bent niet verplicht deze licentieovereenkomst te accepteren, aangezien U deze niet ondertekend hebt. Echter, niets anders geeft U de toestemming om het Programma of werken gebaseerd op het Programma te wijzigen of te verspreiden.
Deze daden zijn door de wet verboden als U deze licentieovereenkomst niet accepteert. Daarom geeft u aan dat door het Programma te verspreiden of te wijzigen, U deze licentie, en al zijn voorwaarden en bepalingen in verband met kopiëren, wijzigen of verspreiden van het Programma, of werken gebaseerd op het Programma, accepteert om dat te kunnen doen.

6. Elke keer U het Programma (of een werk gebaseerd op het Programma) verspreidt, krijgt de ontvanger automatisch een licentie van de originele licentiehouder om het Programma te kopiëren, verspreiden of wijzigen, onderworpen aan deze bepalingen en voorwaarden. U mag de ontvanger geen beperkingen opleggen om de rechten uit te oefenen die hierin bepaald zijn.

7. Als door gevolg van een rechterlijke uitspraak of beweringen van patentenschending of door eender welke andere reden (niet beperkt tot patentenproblemen) U bepalingen worden opgelegd (door rechterlijk bevel, overeenkomst, of op andere wijze) die in tegenspraak zijn met bepalingen in deze licentie, dan sluit dat U niet uit om aan de voorwaarden van deze licentie te voldoen. Als U het Programma niet kan verspreiden en daarbij zowel aan tegelijk de bepalingen van deze licentie als aan andere relevante verplichtingen kan voldoen, dan mag U als gevolg daarvan het Programma helemaal niet verspreiden.
Bijvoorbeeld, als een patent licentieovereenkomst niet zou toestaan dat het programma zonder het betalen van royalty's vrij verspreid mag worden door zij die het Programma direct van U verkrijgen en zij die het indirect door U verkrijgen, dan is de enige manier om zowel daaraan als aan deze licentie te voldoen dat U zich compleet onthoudt van het verspreiden van het Programma.

Als een deel van dit artikel ongeldig wordt geacht, of het kan niet afgedwongen worden onder bepaalde omstandigheden dan is het de bedoeling dat het overwicht van dit artikel van toepassing is. In andere omstandigheden geldt dit artikel volledig.

Het is niet het doel van dit artikel om u er toe aan te zetten om patenten, of andere aanspraken van bezit, te schenden of de geldigheid van zulke aanspraken aan te vechten. Het enige doel van dit artikel is om de integriteit te beschermen van het vrije software verspreidingssysteem, dat wordt toegepast door middel van Publieke Licentie praktijken. Veel mensen hebben royale bijdragen geleverd aan het systeem van vrije software rekenend op de betrouwbaarheid van zijn toepassing. Het is aan de auteur/donor om te bepalen of hij of zij bereidt is om software te verspreiden door middel van een ander systeem en een gelicensieerde kan die keuze niet afdwingen.

Dit artikel is bedoeld om zeer duidelijk te maken wat geloofd wordt een gevolg te zijn van de rest van deze licentie.

8. Als de verspreiding of het gebruik van het Programma gelimiteerd is in bepaalde landen, door patenten of door samenwerking van auteursrechthouders, dan mag de oorspronkelijke auteursrechthouder die het Programma onder deze licentie plaatste een expliciete geografische beperking toevoegen zodat verspreiding enkel toegestaan is in of tussen landen die niet uitgesloten zijn. In dat geval bevat deze licentie de beperking alsof ze in de kern van deze licentie geschreven was.

9. De Free Software Foundation mag gereviseerde en/of nieuwe versies van de Algemene Publieke Licentie uitbrengen van tijd tot tijd. Zulke nieuwe versies zullen gelijkaardig in karakter zijn in vergelijking met de huidige versie maar kunnen in details verschillen om nieuwe problemen of aangelegenheden te behandelen. Elke versie krijgt een expliciet versienummer mee. Als het Programma een versie van deze licentie specificeert waarop het van toepassing is en "elke volgende versie", dan hebt U de keuze om de bepalingen en voorwaarden van die licentie te volgen, of van eender welke versie die later uitgegeven werd door de Free Software Foundation. Als het programma geen versie nummer van de licentie specificeert, dan mag U de bepalingen en voorwaarden volgen van eender welke versie ooit uitgegeven door de Free Software Foundation.

10. Indien U delen van het Programma wil invoegen in andere vrije Programma's welke's verspreidingsvoorwaarden anders zijn, dan moet U de auteur van dat programma om toestemming vragen. Voor software waarvan het auteursrecht bij de Free Software Foundation rust, schrijf naar de Free Software Foundation; we maken hier soms uitzonderingen op. Onze beslissing zal geleid worden door onze twee hoofddoelen om de vrije status van de afgeleiden van onze vrije software te vrijwaren en om het delen en hergebruiken van software in het algemeen te promoten.

11. OMDAT HET PROGRAMMA ZONDER KOSTEN IN LICENTIE GEGEVEN WORDT, IS ER GEEN GARANTIE VOOR HET PROGRAMMA, VOOR ZOVER MOGELIJK BINNEN DE GELDENDE WETGEVING.
UITGEZONDERD WANNEER HET EXPLICIET GESCHREVEN STAAT LEVEREN DE AUTEURSRECHTHOUDERS HET PROGRAMMA "ZOALS HET IS", ZONDER EENDER WELKE GARANTIE, EXPLICIET UITGEDRUKT OF IMPLICIET BEDOELD, ZOALS, MAAR NIET GELIMITEERD TOT, DE IMPLICIETE GARANTIES VAN VERKOOPBAARHEID EN GESCHIKTHEID VOOR EEN BEPAALD DOEL. HET VOLLEDIGE RISICO BETREFFENDE DE KWALITEIT EN DE PRESTATIES VAN HET PROGRAMMA LIGT BIJ U. MOCHT HET PROGRAMMA DEFECT BLIJKEN DAN DRAAGT U DE KOSTEN VAN ALLE BENODIGDE DIENSTEN, REPARATIES OF CORRECTIES.

12. IN GEEN ENKEL GEVAL, TENZIJ VEREIST DOOR DE GELDENDE WET, OF SCHRIFTELIJK OVEREENGEKOMEN ZAL DE AUTEURSRECHTHOUDER, OF EENDER WELKE DERDE PARTIJ DIE HET PROGRAMMA MAG WIJZIGEN EN/OF VERSPREIDEN ZOALS TOEGESTAAN HIERBOVEN, VERANTWOORDELIJK KUNNEN WORDEN GEACHT TEGENOVER U BETREFFENDE ALGEMENE, SPECIALE, UITZONDERLIJKE OF RESULTERENDE SCHADE DIE VOORTVLOEIT UIT HET GEBRUIK, OF DE ONKUNDIGHEID OM HET PROGRAMMA TE GEBRUIKEN
(INCLUSIEF, MAAR NIET GELIMITEERD TOT HET VERLIES VAN GEGEVENS, GEGEVENS DIE CORRUPT WORDEN, OF VERLIEZEN GELEDEN DOOR U OF DERDE PARTIJEN OF EEN FALING VAN HET PROGRAMMA OM SAMEN TE WERKEN MET ANDERE PROGRAMMA'S), ZELFS INDIEN DE AUTEURSRECHTHOUDER OF EEN ANDERE PARTIJ GEÏNFORMEERD WAS OVER DE MOGELIJKHEID TOT ZULKE SCHADE.

EINDE VAN DE BEPALINGEN EN VOORWAARDEN

Hoe deze bepalingen op uw nieuwe Programma's toepassen.

Als U een nieuw Programma ontwikkelt en U wenst dat het van het grootst mogelijk nut is voor iedereen, dan is de beste manier om dit te bereiken door het Programma vrije software te maken dewelke iedereen kan verspreiden en wijzigen onder deze bepalingen.
Om dit te doen, voeg volgende boodschap toe aan het Programma. Het is het veiligst om ze in te voegen aan het begin van elk bronbestand, dit om het ontbreken van garantie duidelijk te maken; en elk bestand

zou minstens de "auteursrecht" lijn en een directief naar waar de volledige boodschap gevonden kan worden moeten bevatten.

<een regel voor de naam van het Programma en zijn doel>
Auteursrecht (C) <jaar> <naam van de Auteur>
Dit Programma is vrije software; U kan het verspreiden en/of wijzigen onder de bepalingen van de GNU Algemene Publieke Licentie, zoals uitgegeven door de Free Software Foundation; oftewel versie 2 van de Licentie,of (naar vrije keuze) een latere versie.
Dit Programma is verspreid met de hoop dat het nuttig zal zijn maar ZONDER EENDER WELKE GARANTIE; zelfs zonder de impliciete garantie van VERKOOPBAARHEID of GESCHIKTHEID VOOR EEN BEPAALD DOEL. Zie de GNU Algemene Publieke Licentie voor meer details. U zou een kopie van de GNU Algemene Publieke Licentie ontvangen moeten hebben samen met dit Programma; indien dit niet zo is, schrijf naar
de Free Software Foundation, Inc.,
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Voeg ook informatie bij hoe men U kan contacteren via e-mail en gewone post. Als het Programma interactief is, laat het een korte boodschap tonen zoals deze wanneer het in interactieve modus start:

Fiscus versie 69, Auteursrecht (C) <jaar> <naam v/d auteur>
Fiscus komt met ABSULUUT GEEN GARANTIE; voor details typ 'toon w'. Dit is
vrije software en het is U toegestaan deze te verspreiden onder bepaalde voorwaarden; typ 'toon c' voor meer details.

U zou ook uw werkgever (indien U als programmeur werkt) of uw school, indien die er is, om een "auteursrecht afwijzing" te laten tekenen voor het Programma, indien nodig. Hier is een voorbeeld; wijzig de namen:

Yoyodyne, NV., verwerpt hier alle auteursrechtlijk interesses in het Programma
Fiscus (dat belastingsaangiften invult) geschreven door James Hacker.
<handtekening van Ty Coon>, 21 April 1984 Ty Coon, Vice voorzitter.

Deze Algemene Publieke Licentie laat niet toe dat het Programma verwerkt wordt in een commercieel programma. Als uw Programma een subroutine bibliotheek is, dan kan U het misschien nuttige beschouwen om toe te staan dat uw Programma gelinkt word met commerciële programma's. Als dat is wat U wil doen, dan moet U de GNU Algemene Minder Publieke Licentie gebruiken in plaats van deze licentie.