Documentatie Grafische Bibliotheek

1 Inleiding

Dit document bevat een beschrijving van de operaties die voorzien worden in de grafische bibliotheek. Het is niet de bedoeling om aanpassingen te maken aan de implementatie van deze grafische bibliotheek.

2 Overzicht

De functies weergegeven in Tabel 1 overlopen de procedures die beschikbaar worden na het inladen van de grafische bibliotheek. Deze top-level functies zijn constructor-functies die bepaalde grafische elementen, geabstraheerd als ADT, aanmaken.

Naam	**Argumenten**	**Signatuur**
make-window	width, height, title	number, number, string → Window
make-tile	width, height, file_path [mask_path]	number, number, [string, [string]][1] → Tile
make-bitmap-tile	file_path, [mask_path]	string, [string] → Tile
make-tile-sequence	tiles	pair[2] → TileSequence

Tabel 1: Procedures beschikbaar na het inladen van de bibliotheek.

3 ADT Window

De constructor om een nieuw venster aan te maken, is `make-window`. De breedte, de hoogte en de titel van het aan te maken venster worden als argumenten genomen. Het resultaat van de oproep is een nieuw window object. Tabel 2 geeft een overzicht van de interface van Window objecten. De `set-background!` boodschap verwacht een string die de kleur van de achtergrond zal veranderen. Op een window kunnen tiles en tile-sequences getekend worden, om controle te hebben welke tile vooraan staat en welke tile achteraan staat gebruiken we layers. Door de `make-layer` boodschap te sturen kan de programmeur een nieuwe layer aanmaken. Dit ADT wordt besproken in sectie 4.

De laatste twee boodschappen `set-key-callback!` en `set-update-callback!` zijn de twee procedures die nodig zijn om je spellus te implementeren. Beiden verwachten een functie als argument. Voor de `set-key-callback!` procedure is dit argument een procedure verantwoordelijk voor het afhandelen van toetsenbordinvoer. Van deze procedure wordt verwacht dat deze twee parameters heeft: status en key. Wanneer deze procedure uitgevoerd wordt, zal `status` gebonden worden aan ofwel `'pressed` (wanneer de toets ingedrukt wordt), ofwel aan `'released` (wanneer de toets losgelaten wordt). De parameter `key` zal gebonden worden aan een value die overeenkomt met het soort toets dat ingedrukt is. Als dit een lettertoets is dan zal dit gelijk zijn aan een karakterwaarde (bijvoorbeeld `#\a` voor de letter "a", of het symbool `'space` voor de spatie-toets[3]). Merk op dat wanneer een toets voor langere tijd ingedrukt

[1] Rechthoekige haakjes worden gebruikt voor optionele parameters aan te duiden.
[2] Dit *pair* stelt een lijst van tiles voor.
[3] Lees de documentatie over events in Racket op deze pagina: `https://bit.ly/2EGZ0yr`

wordt dan kan het zijn dat deze procedure meermaals opgeroepen wordt met `'pressed` voor dezelfde toets.

Voor de `set-update-callback!` procedure is het argument opnieuw een functie, verantwoordelijk voor het verzorgen van de spellogica. Het window ADT beschikt zelf over een spellus van waaruit deze functie bij elke iteratie opgeroepen wordt nadat het window al het tekenwerk afgerond heeft. De meegeven functie is op die manier verantwoordelijk voor een iteratie van de spellus van het Frogger spel. Zelf wordt de functie opgeroepen met één argument *x*, een number die het aantal milliseconden aangeeft dat verstreken is sinds de vorige oproep van de functie. Het is met andere woorden de tijd verstreken sinds de vorige iteratie van de lus.

Deze tijdsinformatie wordt doorgaans gebruikt om ervoor te zorgen dat een spel even snel loopt op verschillende hardware. Het is immers niet de bedoeling dat een spel onspeelbaar wordt op een snellere computer. Concreet kan je deze informatie gebruiken om de snelheid te bepalen waaraan bijvoorbeeld je auto's zich moeten voortbewegen. In elk aangemaakt window zie je in de titel trouwens ook de zogenaamde Frames Per Second (FPS). Dit getal geeft aan hoevaak je spellus-functie opgeroepen wordt per seconde. Indien dit getal zeer laag wordt, betekent dit dat het uitvoeren van één enkele iteratie van je spellus (= één oproep van het functie-argument dat aan de procedure `set-update-callback!`) gegeven is, zeer veel tijd in beslag neemt.

Naam	**Argumenten**	**Signatuur**
set-background!	string	string → ∅
make-layer	∅	∅ → Layer
set-key-callback!	functie	(symbol, any → ∅) → ∅
set-update-callback!	functie	(number → ∅) → ∅

Tabel 2: Window ADT

Onderstaande code geeft aan hoe deze functies gebruikt kunnen worden om een nieuw venster aan te maken van 800x600 pixels met de titel "DEMO". Merk op dat je de functie `make-window` pas kan gebruiken nadat je de grafische bibliotheek hebt ingeladen zoals weergeven op lijn 1.

```
1 (#%require "Graphics.rkt")
2 (define window (make-window 800 600 "DEMO"))
3 ((window 'set-background!) "white")
```

Figuur 1: Het aanmaken bibliotheek geïllustreerd.

4 Layer ADT

Omdat je in een spel een achtergrond hebt (bv. de snelweg) waarboven objecten getekend moeten worden (bv. de auto's), voorziet de bibliotheek zogenaamde layers. Elke layer bevat een set van tekenobjecten die zichzelf op de layer kunnen tekenen. Layers worden getekend in de volgorde waarin ze aangemaakt werden. Dit betekent dat elk tekenobject van de eerste layer achter de tekenobjecten van de tweede layer zal staan. Bovendien zorgt het gebruik van layers ervoor dat onderliggende layers die niet veranderd zijn, niet opnieuw berekend en getekend moeten worden, wat bruikbaar is om de snelheid van het spel optimaal te maken.

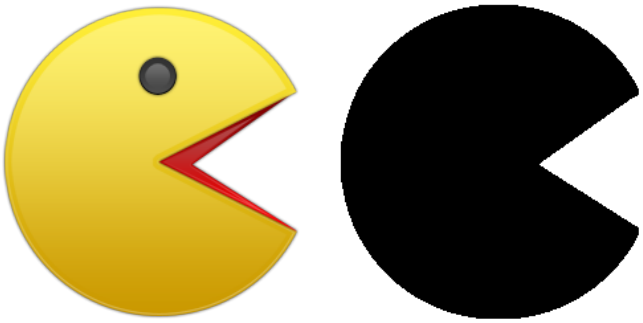
Tabel 3 geeft de interface van het Layer ADT weer. Dit ADT begrijpt de boodschappen `add-drawable` en `remove-drawable` die respectievelijk een tekenelement aan de layer toevoegen en weer verwijderen.

Alleen objecten die minstens de boodschappen `draw` en `set-on-update!` begrijpen, kunnen als tekenobject aan een layer toegevoegd worden. Aan een `draw` boodschap wordt een zogenaamde "drawing

[4] Aangezien een layer ook getekend moet worden ondersteunen layers zelf ook de `draw` boodschap

Naam	Argumenten	Signatuur
add-drawable	drawable	any → boolean
remove-drawable	drawable	any → boolean
draw[4]	draw-context	DrawContext → ∅

Tabel 3: Operaties van het Layer ADT



Figuur 2: Links een bitmap voor Pac Man, rechts diens mask

context" als argument meegegeven. Dit is een intern object van de grafische bibliotheek. Het vereist dus wel wat kennis om zelf een tekenobject te implementeren. Daarom bieden we in de bibliotheek reeds de tekenobjecten Tile en TileSequence aan (zie Sectie 5).

5 Tile & TileSequence ADT

De procedure `make-tile` is een constructor die een Tile aanmaakt, gegeven een width en height en optioneel een pad naar een bitmap (bv. een png of jpg bestand) en een mask.[5] Een mask is een zwart-wit bitmap die door de bibliotheek gebruikt zal worden om te bepalen welke stukken van de bitmap doorzichtig zijn en welke niet. Een mask is een zwart-wit bitmap die door de bibliotheek gebruikt zal worden om te bepalen welke stukken van de bitmap doorzichtig zijn en welke niet. Figuur 2 toont een voorbeeld van een bitmap voor Pac Man en diens mask. Indien je geen mask hebt voor je bitmap, kan je procedure `generate-mask` van de grafische bibliotheek gebruiken om er eentje te genereren. Deze procedure heeft twee argumenten: een pad naar een procedure, en een achtergrondkleur die verdwijnt als de bitmap tile met een mask ingeladen wordt.

Het resultaat van een oproep naar `make-tile` is een instantie van het Tile ADT. Opdat de corresponderende figuur getekend zou worden op een venster, moet de Tile instantie toegevoegd worden aan een Layer instantie. De Tile zal vanaf dan automatisch hertekend worden op het venster als de Tile wordt aangepast. Je kan de tile bijvoorbeeld van positie doen veranderen op het venster met `set-x!` en `set-y!`. Je kan de Tile roteren met `rotate-clockwise` of `rotate-counterclockwise`. Houd er rekening mee dat rotaties enkel werken op vierkante tiles aangezien anders de width en height van de tile verandert. Verder kan je van Tiles ook de hoogte en de breedte en de positie op het venster lezen. Tabel 4 toont de verschillende operaties ondersteund door het Tile ADT.

Je kan ook zelf tekenen op een Tile met functies zoals `draw-rectangle`, `draw-ellipse`, `draw-text` en `draw-line`. Indien je bij het aanmaken van de tile de optionele bitmap en mask meegaf, dan zal je tekenen bovenop de bitmap met deze functies. `clear` maakt de tile leeg indien er geen bitmap werd meegegeven. Anders zal `clear` terug de orginele bitmap tonen. Zo kan je bijvoorbeeld een rood oog op je Pac-Man tekenen en achteraf terug verwijderen. Of je kan dit gebruiken om de wegen van de achtergrond te tekenen op

[5] We voorzien een extra constructor `make-bitmap-tile` die enkel een bitmap neemt en optioneel een mask. Deze constructor gebruikt de hoogte en de breedte van de bitmap voor de achterwege gelaten argumenten.

een lege tile. Houd er rekening mee dat een mask alles buiten de mask kan verborgen houden, inclusief de rechthoeken of ellipsen die je zelf tekent.

Met `make-tile-sequence` maak je een TileSequence aan. Een TileSequence is een aaneenschakeling van Tiles waar je sequentieel door kan lopen, de constructor `make-tile-sequence` verwacht dan ook een lijst van Tiles als argument. Net zoals een Tile zal een TileSequence een enkele bitmap op het venster tekenen, maar je kan via oproepen van `set-next!` en `set-previous!` snel de volgende of vorige tile in de sequentie afbeelden. Deze abstractie is vooral handig om animaties in je spel te incorporeren. In het voorbeeld van Pac Man kan je een animatie gebruiken om Pac Man te laten happen. Dit doe je door een TileSequence aan te maken met twee Tiles. De eerste Tile maak je aan voor een bitmap van Pac Man waarop de mond gesloten is, en de tweede Tile voor een bitmap waar de mond van Pac Man open is. Als je dan snel achter elkaar `set-next!` oproept, zal je Pac Man lijken te happen.

Een TileSequence ondersteunt dezelfde tekenoperaties als een enkele Tile. Hiertoe worden deze operaties gedelegeerd naar de interne lijst van Tiles. Indien je tekent op een TileSequence, wordt dit op alle Tiles in de sequence getekend. Echter, wanneer je tekent op een Tile van een TileSequence, zal het resultaat alleen zichtbaar zijn wanneer de TileSequence toevallig net die Tile aan het afbeelden is. Merk ook op dat het belangrijk is niet én een Tile uit een TileSequence én de TileSequence zelf aan een layer toe te voegen. Anders kan een van beiden verborgen blijven.

Naam	**Argumenten**	**Signatuur**
set-x!	x-coördinaat	number → ∅
set-y!	y-coördinaat	number → ∅
get-x	∅	∅ → number
get-y	∅	∅ → number
get-w	∅	∅ → number
get-h	∅	∅ → number
draw-ellipse	x,y,width,height, color	number, number, number, number, string → number
draw-rectangle	x, y, width, height, color	number, number, number, number, string → number
draw-line	x1, y1, x2, y2, thickness, color	number, number, number, number, number, string → number
draw-text	text, fontsize, x, y, color	string, number, number, number, string → number
rotate-clockwise	∅	∅ → ∅
rotate-counterclockwise	∅	∅ → ∅
draw	dc	DrawContext → ∅
Enkel voor TileSequence:		
set-next!	∅	∅ → ∅
set-previous!	∅	∅ → ∅

Tabel 4: Operaties van de Tile en TileSequence ADT's.