F# quotation to OpenCL translator

или кратко о том, как я пыталась подружить OpenCL и F#

Немного об OpenCL

Пример кода kernel-функции на C++:

```
string sourceString = "\n\
  __kernel void sqr(__global float *matrix, __global float
*res_matr, int matrixSize)\n\
  {\n\
     int  id = get_global_id(0);\n\
     float result = 0;\n\
     int rowInd = id / matrixSize * matrixSize;\n\
     int matrGlobal = matrixSize * matrixSize;\n\
     int colInd = id % matrixSize + id / matrGlobal *
matrGlobal;\n\
     for (int i = rowInd; i < rowInd + matrixSize; i++) {\n\
      result += matrix[i] * matrix[colInd];\n\
      colInd += matrixSize;\n\
     }\n\
     res_matr[id] = result;\n\
  }";
```

Немного об OpenCL

Примерная последовательность запуска*:

- `checkErrorEx( errcode = Platform::get(&platform) );`
- `platform[0].getDevices(CL_DEVICE_TYPE_GPU, &devices);`
- `checkErrorEx( Context context(devices, NULL, NULL, NULL, &errcode) );`
- `checkErrorEx( CommandQueue queue(context, devices[device_index], CL_QUEUE_PROFILING_ENABLE, &errcode) );`
- `checkErrorEx( Program program = Program(context, sourceString, false, &errcode) );`
- `errcode = program.build(devices, "-cl-fast-relaxed-math -cl-no-signed-zeros -cl-mad-enable");`
- `checkErrorEx( Buffer dev_matr = Buffer( context, CL_MEM_READ_WRITE | CL_MEM_COPY_HOST_PTR, arrSize*sizeof(float),  matrix, &errcode ) );`
- `make_kernel< Buffer, Buffer, int > sqr(program, "sqr", &errcode);`
- `  checkError(make_kernel);`
- `EnqueueArgs enqueueArgs(queue, cl::NDRange((arrCount * matrixGlobal + 511) / 512 * 512)/*globalSize*/, NullRange/*blockSize*/);`
- `Event event = sqr(enqueueArgs, dev_matr, dev_res_matr, matrixSize);`

F#: что это?

```
**
let testAnd x y =
    match x y with
    | true, true -> true
    | _ -> false;;
```

```
let books =
    new List<Book>(
        [|
            Book("Programming F#;Cris Smith);
            Book("Foundations of F#;Robert Pickering");
        |]

books.Sort(
    { new IComparer<_> with
        member this.Compare(left:Book, right:Book) =
            left.PublishDate.CompareTo(right.PublishDate)
    }
)
```

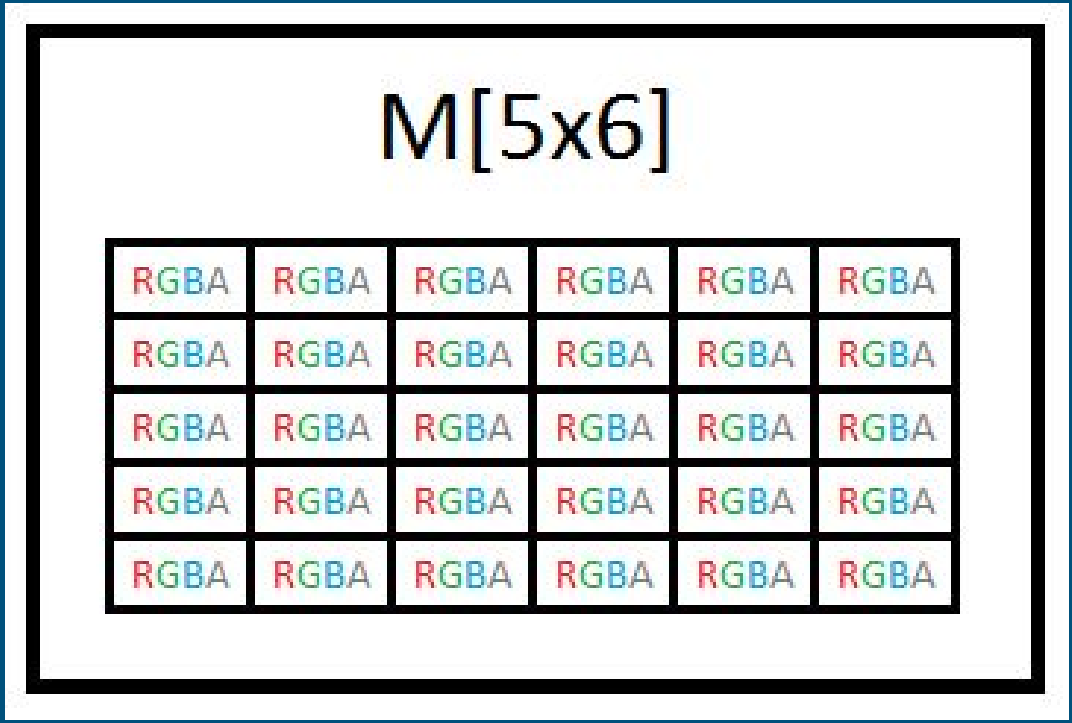
F# -> OpenCL

```
***
let deviceType = DeviceType.Default
let platformName = "NVIDIA*"

let provider =
     try  ComputeProvider.Create(platformName, deviceType)
     with
     | ex -> failwith ex.Message

let checkResult command =
     let kernel,kernelPrepareF, kernelRunF = provider.Compile command
     let commandQueue = new CommandQueue(provider, provider.Devices |> Seq.head)
     let check (outArray:array<'a>) (expected:array<'a>) =
          ...
          let cq = commandQueue.Add(kernelRunF()).Finish()
          let cq2 = commandQueue.Add(outArray.ToHost(provider,r)).Finish()
          commandQueue.Dispose()
          provider.CloseAllBuffers()
     kernelPrepareF,check
```

F# -> OpenCL

```
member this.``Checking of Image2D``() =
    let command =
        <@
            fun (range:_1D) (img:Image2D<ARGB<Float>>) (a:array<_>) ->
                a.[0] <- 1
        @>
    let CLimg = new Image2D<_>(provider, Operations.ReadOnly, true, 10, 10, -1)
    let run,check = checkResult command
    run _1d CLimg intInArr
    check intInArr [|1;3;6;7|]
```

Примитив Image2D



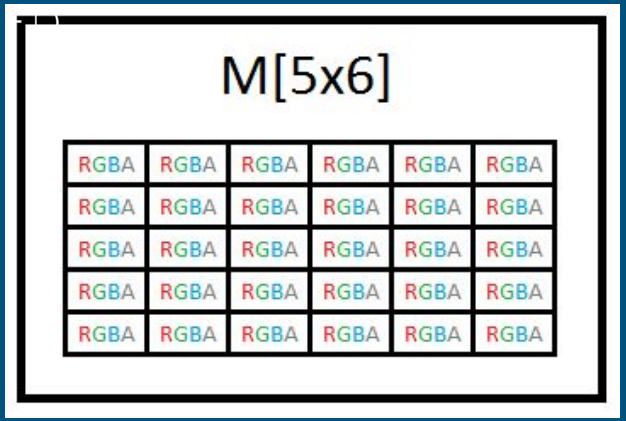
- Snorm_Int8
- Unorm_Short555
- Signed_Int32
- Unsigned_Int16
- Float
- ...

- RGB
- RGBA
- BGRA
- ARGB
- ...

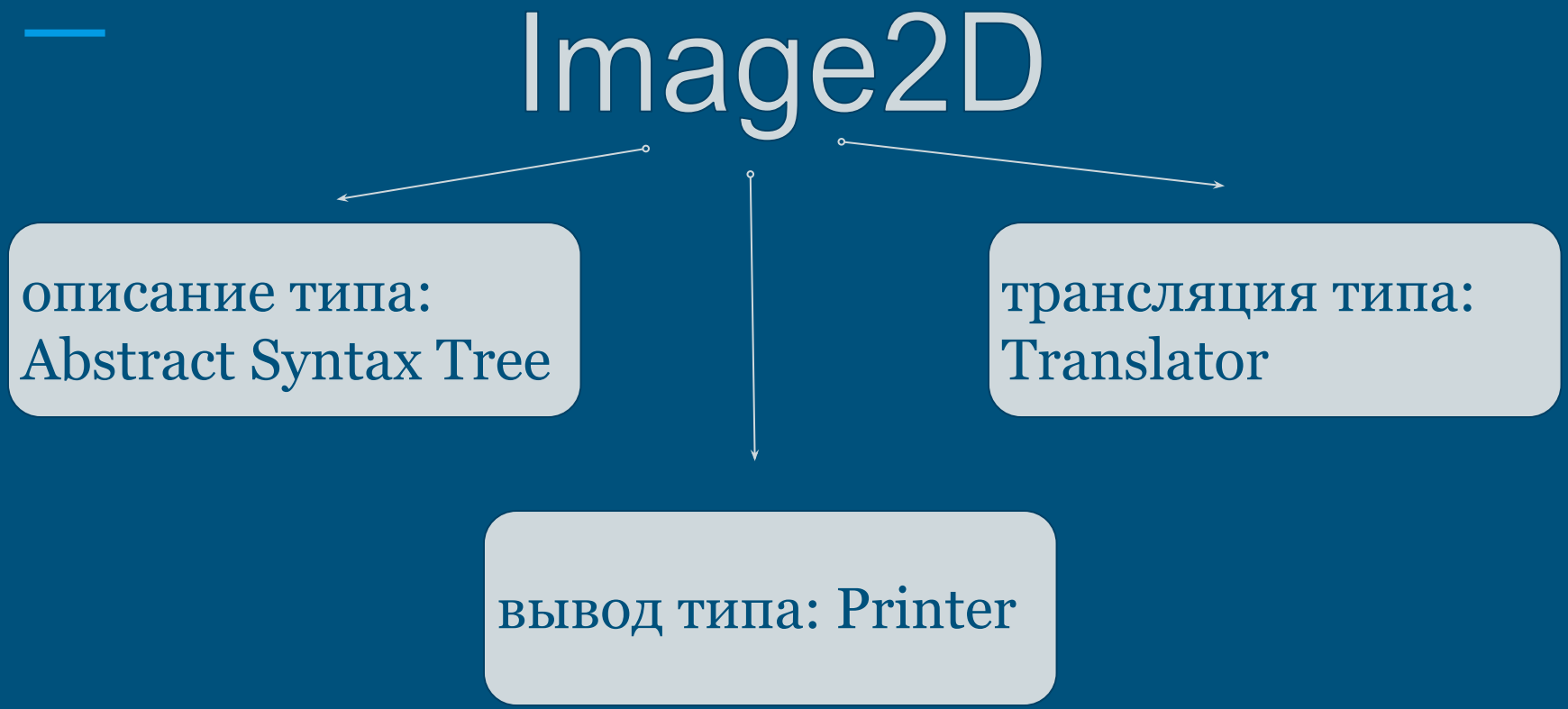
- ReadOnly
- WriteOnly

Примитив Image2D

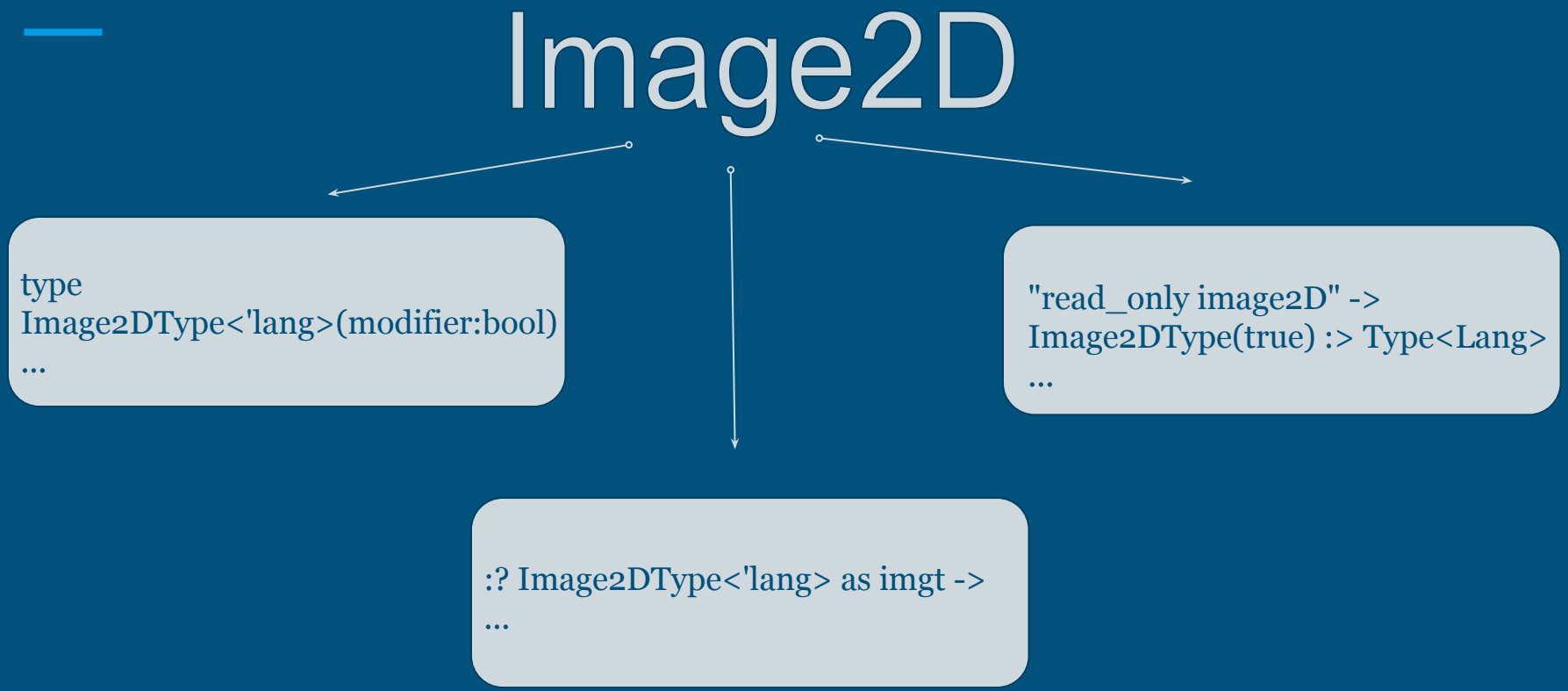
```
public sealed class Image2D<T> : Brahma.Image2D<T> where T : struct,
IImageFormat
    {

    public Image2D(ComputeProvider provider, Operations operations, bool
hostAccessible, int width, int height, int rowPitch = -1);

    public Image2D(ComputeProvider provider, Operations operations, Memory
memory, int width, int height, T[] data, int rowPitch =

    public int Width;
    public int Height;
    public int RowPitch;
    public bool Modifier;
    }
```



Примитив Image2D



Примитив Image2D



Работа с уже реализованными типами

*Проблема****:*

У структуры _1D : INDRangeDimension (Brahma.OpenCL._1D) нет метода size, который позволил бы не передавать длину входного массива цитированной функции.

Варианты решения:

- Если у примитива OpenCL._1D есть метод size, то “подружить” его с транслятором F#
- Проверить возможность использования глобальной переменной size внутри транслятора

Работа с уже реализованными типами

*Проблема****:*

У структуры _1D : INDRangeDimension (Brahma.OpenCL._1D) нет метода size, который позволил бы не передавать длину входного массива цитированной функции.

Решения:

- ...
- Да, глобальные переменные транслятор вполне себе умеет обрабатывать

Ссылки:

* http://edu.mmcs.sfedu.ru/course/view.php?id=225
** http://fprog.ru/2010/issue5/maxim-moiseev-et-al-fsharp-intro/
***
https://github.com/YaccConstructor/Brahma.FSharp/blob/master/tests/Brahma.FSharp.OpenCL.Tests/Full.fs
**** https://github.com/YaccConstructor/Brahma.FSharp/issues/33