Cs-gO Documentation

Fast, portable GPU programming backed by OpenGL Compute Shaders

Team Members
JeanHeyd Meneide (jm3689)
Jett Andersen (jca2136)
David Naveen Dhas Arthur (da2647)

All code can be viewed at https://github.com/daviddhas/CS-gO.

namespace csgo

struct program

```
template<typename F>
program::program(F&& f, const size_list_t& sizes, bool makeContext = false)
```

> `f` is a function whose arguments are all of type `image2d`, and whose output is either an `image2d` or a tuple of `image2d`'s. The vector at position *i* in `sizes` must contain the height and width of the *ith* element of the tuple returned by `f`. If `makeContext` is true, then we create an OpenGL context using GLFW.

```
template<typename... Args>
dsl::io_result program::operator()(Args&&... args)
```

> Each argument in `args` must be an `image2d_io` type. The *ith* argument in `args` must have the same template parameter as the *ith* argument of the function passed to the `program` constructor. The return value is a type that can be cast to a `std::tuple` of `image2d_io` types. Again, the *ith* parameter of the `std::tuple` must have the same template parameter as the *ith* element of the return value from the function passed to the `program` constructor.

template <typename P> struct image2d_io

```
image2d_io::image2d_io(const std::vector<P>& vals, int width))
```

> An `image2d_io` is a wrapper for an OpenGL texture, whose data is stored on the GPU. `P` may be of type `float`, `glm::vec2`, `glm::vec3`, or `glm::vec4`. The texture will be initialized using `vals`, and `width` is the width of the texture. The height is automatically computed using the size of `vals`.

```
image2d_io::image2d_io(texture_data data)
```

Constructs an `image2d_io` directly from an existing OpenGL texture. `texture_data` is a simple record containing only three `GLuint` elements: `id` (the texture handle), `width`, and `height`.

```
std::vector<P> image2d_io::read() const
```

Returns the data stored in the texture on the GPU.

```
GLuint image2d_io::get_texture_id() const
```

Returns the texture handle of the OpenGL texture.

struct display

```
template<typename T>
static void image(const image2d_io<T>& input)
```

Displays the data contained in `input` in the current OpenGL context, by rendering the texture to a full-screen quad.

namespace csgo::dsl

```
template <typename P>
struct image2d
```

The counterpart to `image2d_io` that is operated on using the CS Go dsl (domain-specific language). All operations on `image2d` types return intermediate representations. Any operation that can be applied to `image2d` can also be applied to these intermediate representations, besides the one generated by indexing. The intermediate representations are used by `program` to generate an abstract syntax tree that compiles to the GLSL backend.

```
template <typename T>
indexing image2d::operator[](T&& idx)
```

Returns and expression representing the indexing operation on this image2d type. It can be chained with other expressions.

```
template <typename T>
image_variable& image2d::operator= (T&& right)
```

Generates an assignment operation. Should be treated as normal assignment, but note that it also triggers a statement generation in the underlying code.

```
template <typename L, typename R>
inline addition operator + (L&& l, R&& r)
```

Generates an addition operation expression. `L` and `R` must be expressions. Analogous functions exist for `-`, `*`, and `/`. SFINAE is used to ensure that at least one of L and R are expressionable types.