Enhanced typename

Tian Liao

November 25, 2023

Document number:
Date:
Audience:
Authors: Tian Liao
Reply-to: Tian Liao <tilia@microsoft.com>

1 Introduction

2 Proposal

Quick glance

```
// declare some materials
struct Color { void apply() {} };
struct Texture { void apply() {} };
struct Glass { void apply() {} };

// define a type alias
typename Material { void apply(); };

int main() {
  {
    // use Material as a pointer
    Color color;
    Material* material = color;
    material->apply();
  }
  {
    // use Material as a reference
    Texture texture;
    Material& material = texture;
    material.apply();
  }
  {
    // host a Material in unique ptr
    std::unique_ptr<Material> material{new Glass()};
    material->apply();
  }
}
```

Type alias can also combine with other type aliases to form a new type alias.

```
typename Gettable{ void get(); };
typename Settable{ void set(); };

typename GetSet : Gettable, Settable {};
typename GetSetEquvalent {
  void get();
  void set();
};
static_assert(std::is_same_v<GetSet, GetSetEquvalent>);
```

Type alias can also have specific constraints to control its copiability, relocatability, etc.

```
typename NoCopyNoMove {
  NoTrivial(const NoTrivial&) = delete;
  NoTrivial(NoTrivial&&) = delete;
};

void foo(NoCopyNoMove& a, NoCopyNoMove& b) {
  a = b; // compile error
  a = std::move(b); // compile error
}
```

3 Motivation