

Spike: 7**Title:** Composite & Command Patterns**Author:** Ryan Chessum, 102564760**Goals / deliverables:**

- Code, see /12 – Spike – Composite & Command Patterns /Zorkish Task 13/
- Spike Report

Technologies, Tools, and Resources used:

List of information needed by someone trying to reproduce this work

- Visual Studio 2019
- C plus plus reference (<https://www.cplusplus.com/reference/>)
- Zorkish game specifications
- Code from previous spikes

Tasks undertaken:

- Download and install Visual Studio
- Create a new C++ project
- Import code from previous spikes
- Add attributes for entities
- Add functionality for nested entities
- Add ability to load nested entities from file
- Update look command to look at nested entities
- Add take command to take entities
- Add throw command

What we found out:

In games programming two important patterns to understand are the composite and component patterns.

The composite pattern is where an object is made up of other objects. Some way this could be implemented is having items in a chest. Or you could even have an enemy made up of multiple parts like a robot. Each limb of the robot could have its own health etc.

The component pattern is where you can add components that give new attributes or actions to an entity. For example, you could make items with the consumable component able to be eaten. Then that item could also have components that tell the game what effects the player receives when consuming the item.

I made a version of Zorkish that shows a basic version of both ideas.

First, I imported the version of Zorkish from task 12 since it has everything we need to get started.

Starting with the composite pattern, I wanted to demonstrate it by nesting entities within one another. I added a vector of entity pointers to the entity class which will allow them to hold entities within.

```
vector<Entity*> items;  
int health;  
vector<Attribute> attributes;
```

Next, I created a class for components which I called attributes and added a vector of them to the entity class. Some of the attributes I want to add are “searchable” which means a player will be able to look at the items nested in it, “takeable” meaning an item can be put in the players inventory, “throwable” meaning a player can throw the item at something and “health” meaning the entity will have health.

```
#include "IdentifiableObject.h"  
class Attribute :  
    public IdentifiableObject  
{  
public:  
    Attribute() {};  
    Attribute(vector<string> ids);  
    ~Attribute() {};  
};
```

The components I’m adding here will only be very simple tags for the most part and can be expanded upon in the future to hold more data. The advantage for this implementation is that the components can easily be read from file.

Since the health attribute is the only attribute that has data, I simply added an int to entities. If they have a health component it gets set to 100, otherwise it is left NULL.

```
# comment :)
# L specifies a location
# C specifies a list of connections and directions
# the connection list should be underneath the location as the file will be read in order
# every room should have at least 1 unique identifier to specify which rooms connect to it

L|hallway,starting room|Hallway|It is a dimly lit hallway
E|torch|Torch|it's a torch sitting on the wall
C|north,empty room

L|room,empty room|Empty Room|It is an empty room with a door on each of the four walls
C|south,starting room|north,north room|east,east room|west,west room

L|small room,north room|Small Room|There isn't much here
E|flower patch,flowers|Flower patch|It's a small patch of flowers
E|broken ladder,ladder|Broken ladder|It's a broken ladder
C|south,empty room

L|small room,east room|Small Room|There isn't much here
E|flower patch,flowers|Flower patch|It's a small patch of flowers
E|rock,stone|Stone|It's a small stone|takeable,throwable
E|chest|Chest|It's a chest, what could be inside|searchable
I|chest|thing|Thing|It's a um thing i guess|takeable
E|torch|Torch|it's a torch sitting on the wall
C|west,empty room

L|small room,west room|Small Room|There isn't much here
E|torch|Torch|it's a torch sitting on the wall
E|troll|Troll|A giant fearsome creature|health
I|troll|gem|Gem|It's a bright gem|takeable
I|troll|thing|Thing|It's a um thing i guess|takeable
I|troll|rock,stone|Stone|It's a small stone|takeable,throwable
C|east,empty room
```

I slightly changed how reading in works. All entities and locations are loaded in first. I tried to implement nesting in the same loading phase however if a vectors size is changed the memory location of all the data stored inside changes. So, we have to store them all first before we can start placing them in locations and other entities.

```

fs.open(fileName, fstream::in);

//Get Locations
while (getline(fs, str))
{
    if ((str.size() != 0) && (str.at(0) == 'L'))
    {
        vector<string> details = split(str, '|');

        locations.push_back(Location{ split(details[1], ','), details[2], details[3] });
    }
    if ((str.size() != 0) && (str.at(0) == 'E'))
    {
        if (!locations.empty())
        {
            vector<string> details = split(str, '|');
            if (details.size() > 4)
            {
                entities.push_back(Entity{ split(details[1], ','), details[2], details[3], split(details[4], ',') });
            }
            else
            {
                entities.push_back(Entity{ split(details[1], ','), details[2], details[3], {" "} });
            }
        }
    }
    if ((str.size() != 0) && (str.at(0) == 'I'))
    {
        if (!locations.empty())
        {
            vector<string> details = split(str, '|');
            if (details.size() > 5)
            {
                entities.push_back(Entity{ split(details[2], ','), details[3], details[4], split(details[5], ',') });
            }
            else
            {
                entities.push_back(Entity{ split(details[2], ','), details[3], details[4], {" "} });
            }
        }
    }
}
}

```

After they are loaded in, they are added to their locations on the second loading phase. If 'I' is specified instead of 'E' the entity is placed in the entity in the same location that has the same id specified at the start.

```

fs.open(fileName, fstream::in);

while (getline(fs, str))
{
    vector<string> details = split(str, '|');

    if ((str.size() != 0) && (str.at(0) == 'L'))
    {
        loc++;
    }
    if ((str.size() != 0) && (str.at(0) == 'E'))
    {
        ent++;
        if (!locations.empty())
        {
            locations.at(loc).entities.push_back(&(entities.at(ent)));
        }
    }
    if ((str.size() != 0) && (str.at(0) == 'I'))
    {
        ent++;
    }
}

fs.close();

```

The in the third phase connections are added like before.

Now everything should load in.

I then updated the look command to be able to look at nested entities by searching the entities list of nested entities for a match. To be able to look at the entities nested in another entity it must have a “searchable” component.

```
string LookCommand::Execute(vector<string> input, Location* location, Player* player)
{
    if (input.size() > 1 && input.at(1) == "at")
    {
        if (input.size() > 3 && input.at(3) == "in")
        {
            if (input.size() > 4)
            {
                if (player->AreYou(input.at(4)))
                {
                    if (player->HasItem(input.at(2)))
                    {
                        return player->Fetch(input.at(2))->GetFullDescription();
                    }
                }
                else
                {
                    for (auto e : location->entities)
                    {
                        if (e->AreYou(input.at(4)))
                        {
                            for (auto a : e->attributes)
                            {
                                if (a.AreYou("searchable"))
                                {
                                    for (auto i : e->items)
                                    {
                                        if (i->AreYou(input.at(2)))
                                        {
                                            return i->GetFullDescription();
                                        }
                                    }
                                    return "I cannot find the " + input.at(2) + " in the " + input.at(4);
                                }
                            }
                            return "I cannot look inside the " + e->GetName();
                        }
                    }
                    return "I cannot find the " + input.at(4);
                }
            }
            else
            {
                return "What did you want to look in?";
            }
        }
        if (input.size() > 2)
        {
            if (player->AreYou(input.at(2)))
            {
                return player->GetFullDescription();
            }
            else if (player->HasItem(input.at(2)))
            {
                return player->Fetch(input.at(2))->GetFullDescription();
            }
        }
    }
}
```

Next was the Take command. It works similarly to the look command. Entities can be put into the players inventory if they have a “takeable” component.

Entities with the “searchable” component can also have their nested “takeable” items taken out.

```
// take from
if (input.size() > 2 && input.at(2) == "from")
{
    if (input.size() > 3)
    {
        for (auto e : location->entities)
        {
            if (e->AreYou(input.at(3)))
            {
                for (auto a : e->attributes)
                {
                    if (a.AreYou("searchable"))
                    {
                        for (auto i : e->items)
                        {
                            if (i->AreYou(input.at(1)))
                            {
                                for (auto aa : i->attributes)
                                {
                                    if (aa.AreYou("takeable"))
                                    {
                                        player->Put(e->Take(i->FirstId()));
                                        return "You took the " + i->GetName() + " from the " + e->GetName();
                                    }
                                }
                                return "I cannot take the " + i->GetName();
                            }
                        }
                        return "I cannot find the " + input.at(1) + " from the " + input.at(3);
                    }
                }
            }
            return "I cannot look inside the " + e->GetName();
        }
    }
    return "I cannot find the " + input.at(3);
}
return "What did you want to take the " + input.at(1) + " from?";

//take
for (auto e : location->entities)
{
    if (e->AreYou(input.at(1)))
    {
        for (auto a : e->attributes)
        {
            if (a.AreYou("takeable"))
            {
                player->Put(location->Take(e->FirstId()));
                return "You took the " + e->GetName();
            }
        }
    }
}
```

The last new command I added was the Throw command. If a player has an item with a “throwable” component it can be thrown at an entity with a “health” component in the location. If that entities health reaches 0, they get a “searchable” component meaning players can now look at the nested items.

```
string ThrowCommand::Execute(vector<string> input, Location* location, Player* player)
{
    if (input.size() > 1)
    {
        for (auto p : player->items)
        {
            for (auto a : p->attributes)
            {
                if (a.AreYou("throwable"))
                {
                    if (input.size() > 2 && input.at(2) == "at")
                    {
                        for (auto e : location->entities)
                        {
                            if (e->AreYou(input.at(3)))
                            {
                                for (auto aa : e->attributes)
                                {
                                    if (aa.AreYou("health"))
                                    {
                                        if (e->health != NULL && e->health > 0)
                                        {
                                            player->Take(p->FirstId());
                                            e->health -= 100;
                                            string output = "You threw the " + p->GetName() + " at the " + e->GetName();
                                            if (e->health <= 0)
                                            {
                                                output += "\nThe " + e->GetName() + " fell to the ground...";
                                                e->attributes.push_back(Attribute({ "searchable" }));
                                            }
                                            else
                                            {
                                                output += "\nThe " + e->GetName() + " took some damage";
                                            }
                                            return output;
                                        }
                                        return "the " + input.at(3) + " is already dead.";
                                    }
                                }
                                return "I probably shouldn't throw the " + input.at(1) + " at the " + input.at(3);
                            }
                        }
                        return "I cannot find the " + input.at(3);
                    }
                    return "what did you want to throw the " + input.at(1) + " at?";
                }
            }
        }
        return "Cannot throw the " + input.at(1);
    }
    else
    {
        return "What did you want to throw?";
    }
}
```

After adding them to the command manager everything should run.

```
Select C:\Users\ryanc\OneDrive\Documents\GitHub\GamesProgramming\cos30031-102564760\13 - Spike - C...
west
You can see:
  Flower patch
  Chest
  Torch

Go west
Moving west

You are in Empty Room. It is an empty room with a door on each of the four walls.
You can move:
  east
  north
  south
  west

You can see:
Nothing...

Go west
Moving west

You are in Small Room. There isn't much here.
You can move:
  east

You can see:
  Torch
  Troll

Look at troll
Troll: A giant fearsome creature (health : 100)

You are in Small Room. There isn't much here.
You can move:
  east

You can see:
  Torch
  Troll

Look in troll
You cannot look inside the Troll

You are in Small Room. There isn't much here.
You can move:
  east

You can see:
  Torch
  Troll

Throw rock at troll
You threw the Stone at the Troll
The Troll fell to the ground...

You are in Small Room. There isn't much here.
You can move:
  east








You can see:
  Torch
  Troll

Look in troll
Troll contains:
  Gem
  Thing
  Stone

You are in Small Room. There isn't much here.
You can move:
  east

You can see:
  Torch
  Troll

Look at gem in troll
Gem: It's a bright gem (takeable)
```


		Ryan Chessum	1ce5fc9	Throw command
		Ryan Chessum	c88286a	Task 13 Updated look command with at in pattern
		Ryan Chessum	b5e012f	task 13 take command
		Ryan Chessum	61db825	Task 13 Loading composite entities from file
		Ryan Chessum	91dcbd7	Fixes
		Ryan Chessum	65d2c5f	Task 13 progress
		Ryan Chessum	7357d33	Task 13 Project setup