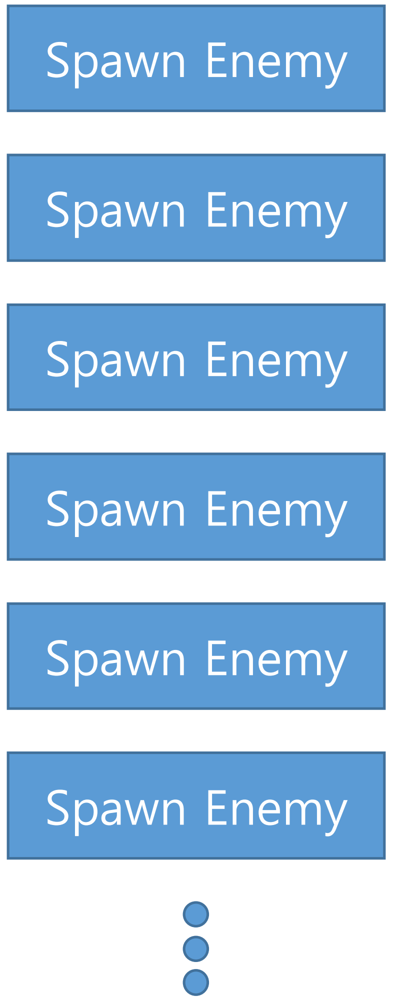
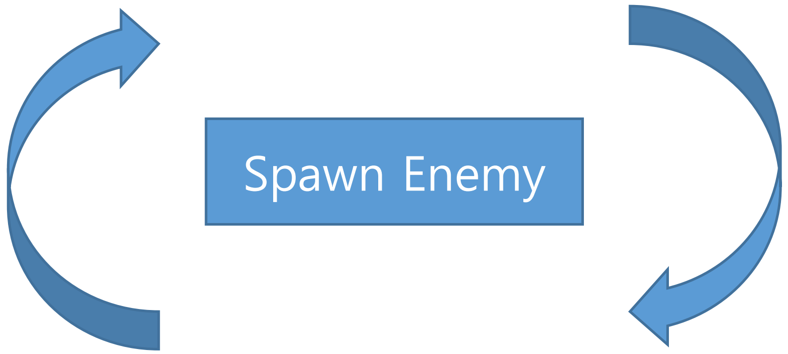
Iteration Statements

DIGIPEN DRAGONS

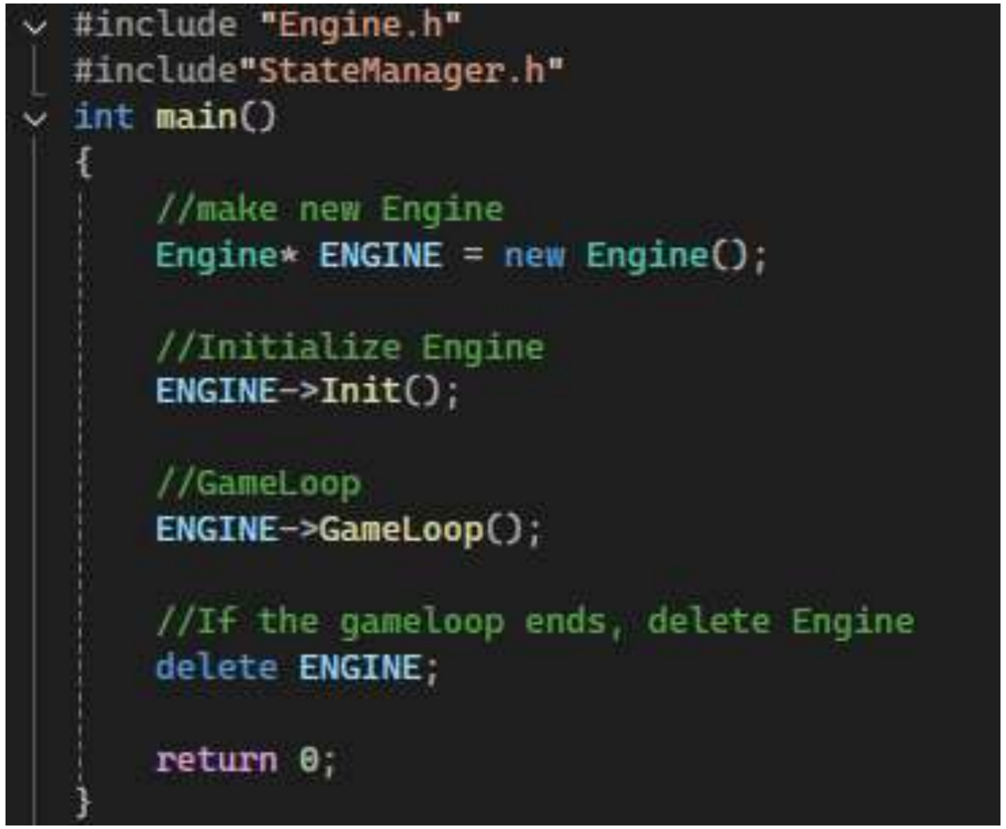
Jinwoo Choi

Iteration Statements(반복문)





Iteration Statements(반복문)



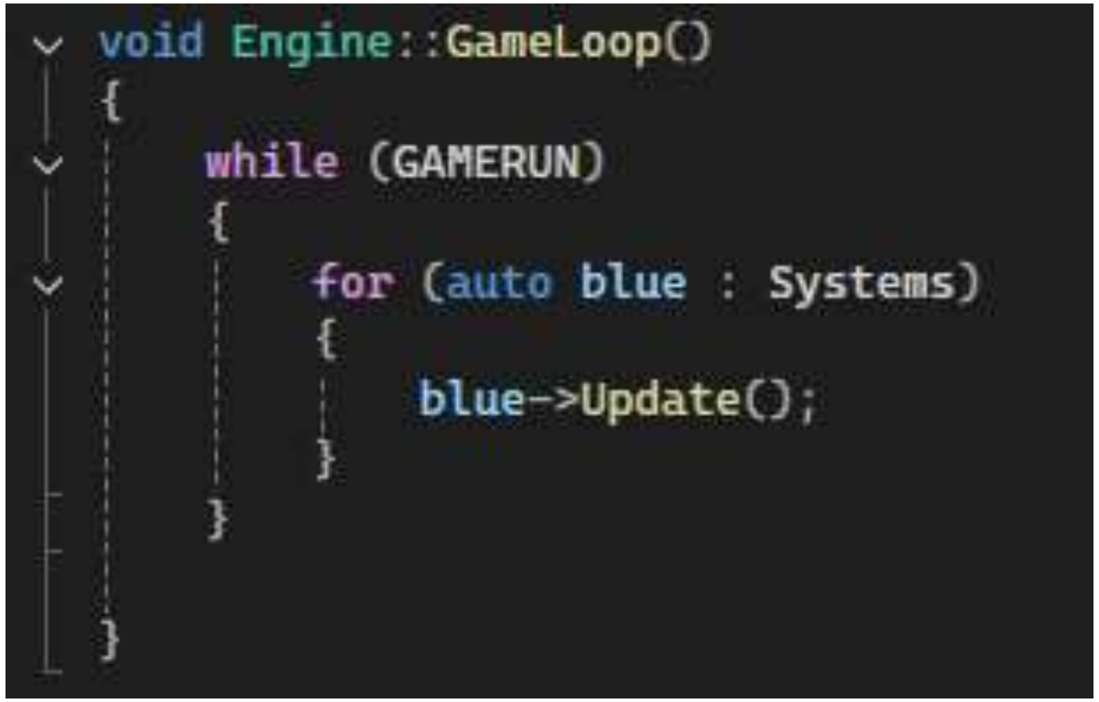
```cpp
#include "Engine.h"
#include"StateManager.h"
int main()
{
    //make new Engine
    Engine* ENGINE = new Engine();

    //Initialize Engine
    ENGINE->Init();

    //GameLoop
    ENGINE->GameLoop();

    //If the gameloop ends, delete Engine
    delete ENGINE;

    return 0;
}
```



```cpp
void Engine::GameLoop()
{
    while (GAMERUN)
    {
        for (auto blue : Systems)
        {
            blue->Update();
        }
    }

}
```

while

```
while(condition)
{
    //executes repeatedly as long as condition is true
}
```

The while loop executes a block of code as long as its condition evaluates to true(non-zero)

And stops execution when the condition becomes false(zero)

do-while

```
do
{
    //executes at least once
    //executes repeatedly as long as condition is true
} while(condition);
```

The do-while loop executes a block of code at least once,

and keeps executing repeatedly as long as its condition evaluates to true(non-zero)

And stops execution when the condition becomes false(zero)

while vs do-while

```
int i = 0;

while(i<15)
{
    printf(“i value: %d\n”, i);
    i++;
}
```

```
i value: 0
i value: 1
.
.
.
i value: 13
i value: 14
```

while vs do-while

```
int i = 0;

do
{
    printf("i value: %d\n", i);
    i++;
}while(i<15);
```

```
i value: 0
i value: 1
.
.
.
i value: 13
i value: 14
```

Infinite Loop

```
while(true)
{
    printf(“infinite loop\n”);
}
```

```
do
{
    printf(“infinite loop\n”);
}while(1);
```

```
int isRunning = 1;
while(isRunning)
{
    if(key_ESC_Pressed)
    {
        isRunning = 0;
    }
    printf(“game loop\n”);
}
```

Control statements

```
while(1)
{
    if(key_ESC_Pressed)
    {
        break;
    }
    printf(“game loop\n”);
}
```

```
while(1)
{
    if(shouldSkip)
    {
        continue;
    }
     printf(“game loop\n”);
}
```

LAB – Sum

- Create a file named 'Sum_YourName.c'.
- Your program should calculate the sum from 1 to 100.
- You must use a while loop to sum the numbers from 1 to 50.
- You must use a do-while loop to sum the numbers from 51 to 100.
- Print each iterator value for every loop iteration and display the final sum on the console.

for

```
for(initialization; condition; update)
{
    //executes repeatedly as long as condition is true
}
```

The for loop executes a block of code as long as its condition evaluates to true(non-zero)

And stops execution when the condition becomes false(zero)

for

```
int count = 10;

for(int i = 0; i < count; i++)
{
    printf(“i value: %d\n”, i);
}
```

```
int count = 10;
int i = 0;

for( ; i < count; )
{
    printf(“i value: %d\n”, i);
    i++;
}
```

```
for( ; ; )
{
    if(key_ESC_Pressed)
    {
        break;
    }
 printf(“infinite loop\n”);
}
```

while vs for

```
int i = 0;

while(i<15)
{
    printf(“i value: %d\n”, i);
    i++;
}
```

```
for(int i = 0; i<15; i++)
{
    printf(“i value: %d\n”, i);
}
```

LAB – SumOdds

- Create a file named 'SumOdds_YourName.c'.
- Your program should calculate the sum of **only odds** from 1 to 100.
- You must use a for loop to perform the calculation.
- Print each iterator value for every loop iteration and display the final sum on the console.

LAB – Password

- Create a file named 'Password_YourName.c'.
- Your program should continuously prompt the user to enter a password.
- You must use a while loop to keep asking for input until the correct password is entered.
- The correct password should be predefined in the program (e.g.,1234)
- Display a success message when the correct password is entered and terminate the program.