```cpp
#include "stdafx.h"

#include "Action.hpp"
#include "InputManager.hpp"
#include "DrawManager.hpp"

#include "ServiceLocator.hpp"

#include "Positions.h"

#include "Entity.h"
#include "Character.h"
#include "Enemy.h"
#include "Item.h"

#include "Typing.h"


Typing::Typing()
{
    m_posInText = 0;
    m_wichText = -1;
}


Typing::~Typing()
{
}

void Typing::Initialize(const std::string& text, const std::string& text1, const std::string& text2,
    float textPos)
{
    DrawManager* drM = ServiceLocator<DrawManager>::GetService();
    Positions* m_pos = ServiceLocator<Positions>::GetService();

    if (!font.loadFromFile("../assets/Font/gamer.ttf"))
    {
        // note(tommi): something went really wrong, we should find out why before proceeding
        // note(tommi): we also want to log this somehow, probably with Debug class
        // Debug::FatalError("...");
        sf::err();

        assert(false);
    }

    m_Textus[0].setFont(font);
    m_Textus[0].setColor(sf::Color::Cyan);
    m_Textus[0].setCharacterSize(24);
    m_Textus[0].setPosition(m_pos->GetText(0).x, m_pos->GetText(0).y);
    m_Textus[0].setString(text);

    m_Textus[1].setFont(font);
    m_Textus[1].setColor(sf::Color::Green);
    m_Textus[1].setCharacterSize(24);
    m_Textus[1].setPosition(m_pos->GetText(1).x, m_pos->GetText(1).y);
    m_Textus[1].setString(text1);

    m_Textus[2].setFont(font);
    m_Textus[2].setColor(sf::Color::Magenta);
    m_Textus[2].setCharacterSize(24);
    m_Textus[2].setPosition(m_pos->GetText(2).x, m_pos->GetText(2).y);
    m_Textus[2].setString(text2);




}

bool Typing::checkLetter()
{
    InputManager* input = ServiceLocator < InputManager>::GetService();


```

```
73      std::string st = "";
74      char key = input->GetLetter() + 65;
75
76      if (m_wichText == -1)
77      {
78
79
80          for (int i = 0; i < 3; i++)
81          {
82              if (key == m_Textus[i].getString()[0] )
83              {
84                      m_wichText = i;
85                      break;
86              }
87          }
88      }
89      else
90      {
91          if (key == m_Textus[m_wichText].getString()[m_posInText])
92          {
93
94              m_Textus[m_wichText].setColor(sf::Color::Black);
95              m_posInText++;
96
97              if (m_posInText == m_Textus[m_wichText].getString().getSize())
98              {
99                  m_Textus[m_wichText].setColor(sf::Color::White);
100                 m_Textus[m_wichText].setString(st);
101
102                 return true;
103             }
104
105
106         }
107
108     }
109     return false;
110 }
111
112 void Typing::newWord(std::string str)
113 {
114
115     m_Textus[m_wichText].setString(str);
116     m_posInText = 0;
117     m_wichText = -1;
118
119 }
120
121 void Typing::TextDraw()
122 {
123     DrawManager* dr = ServiceLocator<DrawManager>::GetService();
124
125     for (int i = 0; i < 3; i++)
126     {
127         if (i == m_wichText)
128         {
129             sf::Text t = m_Textus[m_wichText];
130             sf::Text t2 = m_Textus[m_wichText];
131             std::string str = "";
132             std::string str2 = "";
133
134             for (unsigned int i = 0; i < m_Textus[m_wichText].getString().getSize(); i++)
135             {
136                 if (i < m_posInText)
137                 {
138                     str += m_Textus[m_wichText].getString()[i];
139                 }
140                 else
141                 {
142                     str2 += m_Textus[m_wichText].getString()[i];
143                 }
144             }
145
```

```
146             t.setString(str);
147             t.setColor(sf::Color::Black);
148 
149             t2.setString(str2);
150             t2.setPosition(t.getPosition().x + t.getGlobalBounds().width, t2.getPosition().y);
151             t2.setColor(sf::Color::White);
152 
153             dr->Draw(t, sf::RenderStates::Default);
154             dr->Draw(t2, sf::RenderStates::Default);
155 
156         }
157         else
158         {
159             dr->Draw(m_Textus[i], sf::RenderStates::Default);
160 
161         }
162     }
163 
164 
165 
166 }
167 
```