---

1. Wprowadzenie

Leafy to prosty framework GUI oparty na Raylib, stworzony do łatwego tworzenia nowoczesnych interfejsów w C++. Jego celem jest maksymalna prostota, dzięki której w pliku `demo.cpp` można pisać GUI jak w skrypcie.

---

2. Struktura projektu

```
leafy_demo/
├── include/
│   └── leafy.h       # deklaracje komponentów i layoutów
├── src/
│   └── leafy.cpp     # implementacja
└── demo.cpp          # Twój plik główny z GUI
```

- `include/leafy.h` - nagłówki z API
- `src/leafy.cpp` - implementacja funkcji
- `demo.cpp` - główny plik, w którym piszesz GUI

---

3. Layouty

3.1 BeginVertical / BeginHorizontal

```
leafy::BeginVertical(Rectangle bounds, float spacing = 8, int alignment = 0);
leafy::BeginHorizontal(Rectangle bounds, float spacing = 8, int alignment = 0);
leafy::EndLayout();
```

- `bounds` - prostokąt określający obszar layoutu
- `spacing` - odstęp między elementami
- `alignment`:
- `0` - lewo/góra
- `1` - wyśrodkowane
- `2` - prawo/dół

3.2 BeginCenter

```
leafy::BeginCenter(float width, float height, float spacing = 12);
```

- Tworzy centralny layout, pionowo i poziomo wyśrodkowany w oknie
- Automatycznie ustawia wszystkie komponenty w pionie
- Nie trzeba ręcznie podawać `Rectangle` w komponentach

---

4. Komponenty

4.1 Button

```
bool leafy::Button(const char* text);
```

- Automatyczny prostokąt (`DEFAULT_WIDTH x DEFAULT_HEIGHT`)
- Zwraca `true` , jeśli kliknięto przycisk
- Tekst automatycznie wyśrodkowany w przycisku
- Przykład:

```
if(leafy::Button("Play")) {
    std::cout << "Play clicked
";
}
```

4.2 TextInput

```
bool leafy::TextInput(std::string &value, int maxLength = 32);
```

- Pole tekstowe, wpisywanie znaków i obsługa backspace
- Zwraca `true` po naciśnięciu `Enter`
- `maxLength` - maksymalna długość wpisanego tekstu
- Przykład:

```
std::string name;
if(leafy::TextInput(name)) {
    std::cout << "Wpisano: " << name << std::endl;
}
```

---

5. Zasady korzystania z API

1. **Nie podawać** `Rectangle` **ręcznie** w Button / TextInput, użyj domyślnych wymiarów lub layoutów
2. **Zawsze kończ layout** wywołaniem `EndLayout()`
3. **Gniazdowanie layoutów** jest dozwolone:

```
leafy::BeginCenter(300,300);
    leafy::BeginHorizontal({0,0,300,50});
        leafy::Button("Yes");
        leafy::Button("No");
    leafy::EndLayout();
leafy::EndLayout();
```

4. Poziome w pionowych i vice versa

5. **Używanie BeginCenter** upraszcza centrowanie w pionie i poziomie

---

6. Przykład pełnego demo.cpp

```
#include "raylib.h"
#include "leafy.h"
#include <iostream>

int main() {
    InitWindow(800,600,"Leafy GUI Demo");
    SetTargetFPS(60);

    std::string name;

    while(!WindowShouldClose()) {
        BeginDrawing();
        ClearBackground(leafy::theme.bg);

        leafy::BeginCenter(250,250);

            if(leafy::Button("Click Me"))
                std::cout << "Button clicked!
";

            if(leafy::TextInput(name))
                std::cout << "Entered: " << name << std::endl;

            if(leafy::Button("Exit"))
```

```
                CloseWindow();

        leafy::EndLayout();

        EndDrawing();
    }

    CloseWindow();
}
```

---

7. Wskazówki

- Każdy layout **automatycznie ustawia pozycję komponentów**
- **Nie mieszaj ręcznych współrzędnych z layoutami**, używaj domyślnych funkcji
- Możesz tworzyć **wiele layoutów w jednym oknie**, aby grupować komponenty
- Layouty są **gnieżdżone**, więc można tworzyć struktury typu panel → przyciski → input

---

8. Podsumowanie

Leafy pozwala w prosty sposób:

- Tworzyć przyciski i pola tekstowe
- Używać layoutów pionowych, poziomych i centrowanych
- Pisać GUI w `demo.cpp` jak skrypt, bez ręcznego pozycjonowania
- Szybko prototypować interfejsy, zachowując estetyczny dark theme

---

Koniec dokumentacji