TexDR 인수인계

> 작성자 : 신승민

소스코드 관리

SVN : file://211.238.177.230/cbuilder_source/TexTricot4.0

reversion : 207

담당 업무 개요

필자가 TexDR에 투입된 것은 중기청 융복합 과제 마무리 6개월 전이다. 당시 프로젝트의 상황은 TexTricot의 외관과 기능을 **C# Winform**으로 포팅을 어느정도 해놓은 상황이였다. 하지만 핵심 기능들에서 오류가 발생하고 누락된 상태였다.

필자가 참조한 문서는 다음과 같다.

```
./TexDR/TexTricot_.pdf
```

김태훈 사원의 인수인계 문서 초기 **프로젝트 설계에 대한 문서**이다. 필자가 제일 참조를 많이한 부분이며 여기서 **최종본과 개념상 큰 차이는 없다**.

```
./TexDR/2017.12.28__.pdf
```

신성주 사원의 인수인계 문서 **초기 시뮬레이션 개발과 관련된 문서**이다. DirectX 를 사용해서 개발을 진행했으니 **Unity를 사용한 현재와는 거리가 있다**.

Solution 분석 및 Project 역할

Solution은 4개의 Project로 구성된 상태다.

- **TexTricotCore**
  - C++
  - 실제 프로젝트 데이터 저장
  - 데이터 변형 및 가공
  - *TCCore namespace*
- **TexTricotCLR**
  - C++ CLI
  - TexTricotCore를 C#에서 사용하기 위해 사용
  - *TCClr namesapce*
- **TexTricot**
  - C# Winform
  - 화면에 보여지는 UI
  - Manager
    - 유틸성 코드들
- **TexTricotControls**
  - 공통적으로 사용할 것 같은 **폼 및 컨트롤을 모아놓은 라이브러리**

Project 분석

TexTricotCore

- **Chainlink.h**
  - Course

```
struct Course
        {
                // data
                int X1, X2;                     // in, via

                // setting
                int mmrack = 1700;
                int runin = 100;
                bool raise = false;
                bool force = false;
```

```
                // treatment
                bool sharding = false;
                bool looping = false;
        };
```

- Chainkink

```
struct Chainlink
        {
                std::wstring el_str;
                std::vector<Course> courses;
                int auto_calc_mode;

                std::vector<int> value_defaults;

                void resize(int course_cnt);

                Course& getCourseClamp(int index);
                const Course& getCourseClamp(int index) const;
        };
```

- Chainlinks
  - Chainlink 리스트를 관리하는 클래스

```
int getChainlinkCounts() const;
int getCourseCount() const;
void setCourseCount(int course_cnt);

int addChainlink();
int addChainlink(int index);
void removeChainlink(int index);

void setInvertChainlink();
ChainlinkPtr getInvertedChainlink(int index);
const ChainlinkPtr getInvertedChainlink(int index) const;
```

- **Guidebars.h**
  - BarType, BarFace, BarEntry 정의
  - GuideBars
    - 가이드바 리스트를 관리
    - 바를 정보를 받아오거나 변경하는 작업을 처리한다.

- **Production.h**
  - Fabric

```
struct Fabric
	{
		std::wstring name;
		std::wstring code;
		std::wstring department;
		std::wstring date;

		float testweight = 0;
		float testwidth = 0;
		float testheight = 0;
		int cutyard = 0;

		std::wstring description;
	};
```

- MachineInfo

```
struct MachineInfo
	{
		std::wstring name;

		float runtime = 0;
		int efficiency = 0;
		int rpm = 0;
		int quantity = 0;
	};
```

- PaymentInfo
  - cst 파일에 필요한 데이터들 저장
- Production
  - Fabric, Machine, Payment 를 전부 관리하는 클래스

- **Project.h**
  - PrjectType
    - **Tricot**, Raschel, RaschelJacquard, **DoubleRaschel**
  - Environment

```
struct Environment
	{
		Environment(ProjectType type)
			: type(type), course(54), wpi(28), cpc(28.0f), width(42.0f)
		{
		}

		const ProjectType type;
		int course;
		int wpi;
```

```
                float cpc;
                float width;
        };
```

MainForm에서 우측에 보여지는 데이터들이다.

- SimulationEnvironment

```
struct SimulationEnvironment
        {
                float   fTexWidth;
                float   fTexCPC;
                float   fTexZoom;
                int             nFuzz;
        };
```

- Project
  - 지금 까지 있던 정의한 자료형들을 전부 관리한다.
  - Chainlinks
  - Environment
  - GuideBars
  - Yarns
  - Production
  - SimulationEnvironment

- **TricotBar.h**
  - TricotCommonData

```
struct TricotCommonData
        {
                int  complete_needle_Front;
                int  complete_needle_Back;
                int  complete_needle_pile;
                bool auto_calc;
        };
```

complete needle 계산할 때 필요하다.

- TricotThreadInfo

```
struct TricotThreadInfo
        {
                bool thread_in;
                int  count;
                int  yarn_index;
```

```
                bool transparent;
        };
```

in/out 세팅 및 thread 세팅하는 곳에서 보여지고 수정되는 데이터

- CapacityInfo

```
struct CapacityInfo
        {
                int use;
                int beam;
                int totaluse;

                double MPY;
                double GPY;
                double percent;
                int mmPerRack;
        };
```

실 사용량 데이터이다. 작업지시서 작성에 필요하다.

- CalcInfo

```
struct CalcInfo
        {
                int formula;
                bool noTension;
                bool showRunin;
        };
```

실 최적화 값 관련된 값이다.

- TricotBar
  - 바 하나가 가지고 있는 Capacity 정보, Calc 정보를 가지고 있다.

- **Yarn.h**
  - 실에 대한 추상 클래스이다.

TexTricot.Form

- ChainlinkGuideBarForm
  - Bar 마다 **Threading 정보와 Course Setting 을 변경** 할 수 있다.

- ChainlinkTextForm
  - Bar 에 **Design 정보를 Text 입력**으로 준다. Mode2 를 주로 사용

        - 12 23 32 ⇒ 1에서 나가고 2로 들어오는 실, 2에서 나가고 3으로 들어오는 실, 3에서 나가고 2로 들어오는 실...

- CompositionForm
    - 실 사용량에 대한 정보를 적어놓을 수 있다.
        - TexTricotCore의 Capacitiy Info 를 변형

TexTricot.Manager

- FileManager
    - ftdr 파일은 기존 xml로 저장한 프로젝트를 Zip 한 형태로 다시 저장하는 것이다.
    - **OpenProject()** 를 사용해 파일시스템에 있는 프로젝트를 띄워준다.
    - **SaveProject()** 를 사용해 현재 작업중인 프로젝트를 원하는 파일 포맷으로 저장한다.
- FormManager
    - 커스텀 Form 을 제작하고 사용하기 위해선 **FormManager.CreateWindowForm() 에 등록**을 해주어야 한다.
    - OpenForm, CloseForm, ToggleForm 을 사용해 Form을 제어한다.
- PrintManager
    - 작업지시서 출력을 위해 변수들을 곧곧에서 받아서 초기화 해놨다.
      (InitMemeberVariable())
    - 해당 데이터들을 사용해서 출력하는 함수를 만들자.
      (SimplePrint(PrintPageEventArgs e))
    - 그리고 printDocument 에서 호출해준다.

시뮬레이션 연결

```
private Process simulationProgram;

2 references
private void simulationToolStripMenuItem_Click(object sender, EventArgs e)
{
    // 시뮬레이션 버튼을 누를때 유니티 시뮬레이션 프로그램이 켜져야한다. - 신승민
    FileManager.Inst.SaveXML(FileManager.SIM_PROPERTY_PATH);
    if (ProjectManager.Inst.Jacquard != null)
    {
        ProjectManager.Inst.Jacquard.Save(FileManager.SIM_JACQUARD_PATH);
    }

    ProcessStartInfo psi = new ProcessStartInfo
    {
        FileName = FileManager.SIM_PROGRAM_PATH
    };

    try
    {
        if (simulationProgram == null)
        {
            simulationProgram = new Process();
            simulationProgram.StartInfo = psi;
            simulationProgram.EnableRaisingEvents = true;
            simulationProgram.Exited += SimulationProgram_Exited;
            simulationProgram.Start();
        }
    }
    catch (Exception)
    {
        simulationProgram = null;
        MessageBox.Show("fail");
    }
}

2 references
private void SimulationProgram_Exited(object sender, EventArgs e)
{
    simulationProgram.Exited -= SimulationProgram_Exited;
    simulationProgram = null;
}
```

ftdr 파일 구성

```
// Environment
XmlElement environment = xmlDoc.CreateElement("environment");
project.AppendChild(environment);
SaveXML_Environment(environment, xmlDoc);

// Yarns
XmlElement yarns = xmlDoc.CreateElement("yarns");
project.AppendChild(yarns);
SaveXML_Yarns(yarns, xmlDoc);

// GuideBar
XmlElement guideBar = xmlDoc.CreateElement("guidebar");
project.AppendChild(guideBar);
SaveXML_GuideBar(guideBar, xmlDoc);

// Chainlinks
XmlElement chainllinks = xmlDoc.CreateElement("chainlinks");
project.AppendChild(chainllinks);
```

```
SaveXML_Chainlinks(chainllinks, xmlDoc);


// SimulationInfo
XmlElement simulationInfo = xmlDoc.CreateElement("simulationInfo");
project.AppendChild(simulationInfo);
SaveXML_SimulationInfo(simulationInfo, xmlDoc);

// Production
XmlElement production = xmlDoc.CreateElement("production");
project.AppendChild(production);
SaveXML_Production(production, xmlDoc);
```

FileManager 에 존재

1. Xml 생성 (SaveXML, SaveFTDR)
2. zip 파일로 만든후 저장 (SaveFTDR)

trc to ftdr

기존 TexTricot의 trc 데이터도 TexDR에서 켜졌으면 좋겠다는 생각에서 시작된 작업, 단순히 trc 데이터를 가져오는 작업만 진행되었다.

기존 Tex시리즈 데이터를 가져오는데 문제가 있다.

> 레거시 데이터는 복잡한 바이너리 파일이다.

따라서 필자가 나름 정리한 trc 파일 구조가 있으나. 실제로 가장 좋은것은 C++ 빌더를 활용해서 trc Save, LoadFromFile 함수 코드를 보는 것이 제일 좋다.

TexTricot C++ 코드에서 ***TMain::TRCSave(AnsiString FileName, TCompressMethod cm)*** *함수를 참조 제작하였다.*

사용 io 함수 : ReadFile, WriteFile 함수

FileManager OpenTRC 함수에 구현해 놓았다. TexTricot 코드를 그대로 순서대로 구현한 함수이니 C++ 코드를 참조하면서 보면 도움이 될것

대체 io 함수 : BinaryReader ReadInt, ReadBytes ....

미결사항

- 이미지를 읽어오는 코드에서 문제가 발생하여 DesginForm, CostForm 의 정보를 받아오는 쪽은 제작을 안하였다. 시뮬레이션에 필요한 정보는 전부 넘어온다.

필자 작업 방식

상황 : **새로운 데이터를 추가해야하는 상황**

1. TexTricot에 **필요한 자료형을 TexTricotCore에 추가한다. (header, source file)**
2. 관련된 **C++/CLI 코드를 TexTricotCLR 제작한다. (header, source file)**
3. 해당 데이터를 **사용하는 Action 을 제작한다.**
4. **TexTricot에서 ActionManager를 사용해 데이터를 조회, 변경한다.**

결론

더블라셀 업체에 테스트 요청해서 받은 것들이 있다.

```
./TexDR/190313___.xlsx
```

추가적으로 김태훈 사원의 초기 설계 문서를 참조를 많이하며 개발하였기 때문에 큰 틀을 벗어나지 않았다. 유지보수시에 해당 문서와 이 문서를 참조해주면 좋겠다.