Rapid-I vision measuring System Project

Internship activity - <Day 35>

Author:Yeswanthram R

**Part program:**

**Part program Load:**

RW\_PartProgram::PartProgram\_Load(System::String^ FilePath)

{

try

{

MAINDllOBJECT->ShowHideWaitCursor(1);

RWrapper::RW\_MainInterface::MYINSTANCE()->WriteUserLog("PartProgram\_Load", "Load PartProgram", "Program Name List");

PPCALCOBJECT->PPAlignEditFlag(false);

this->OneShot\_RefImage\_Loaded = false;

PPCALCOBJECT->IntersectionPointMissed(false);

if(MAINDllOBJECT->getActionsHistoryList().getList().size() > 2)

{

if(MAINDllOBJECT->DemoMode() || MAINDllOBJECT->ShowMsgBoxString("RW\_PartProgramPartProgram\_Load01", RapidEnums::MSGBOXTYPE::MSG\_YES\_NO, RapidEnums::MSGBOXICONTYPE::MSG\_QUESTION))

{

LoadProgram(FilePath);

}

else

RWrapper::RW\_MainInterface::MYINSTANCE()->InternalClearAll(2);

}

else

{

LoadProgram(FilePath);

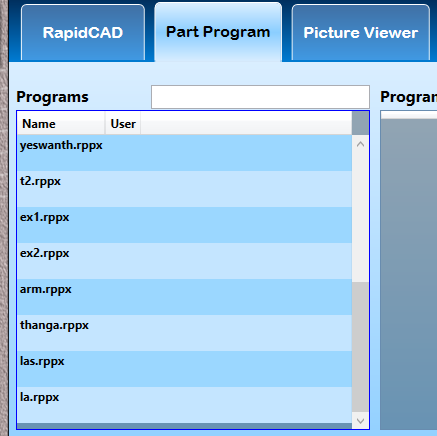
}

MAINDllOBJECT->ShowHideWaitCursor(2);

}

catch(Exception^ ex){ RWrapper::RW\_MainInterface::MYINSTANCE()->WriteErrorLog("WRPP0009", ex); }

}

****

**Part program Build:**

bool RWrapper::RW\_PartProgram::PartProgram\_Build(System::String^ FilePath)

{

try

{

MAINDllOBJECT->SetStatusCode("RW\_PartProgram19");

MAINDllOBJECT->AddAction\_PPEditMode = false;

MAINDllOBJECT->ShowHideWaitCursor(1);

RWrapper::RW\_MainInterface::MYINSTANCE()->WriteUserLog("PartProgram\_Build", "Build PartProgram", "PartProgram Tab");

if(PPCALCOBJECT->IsPartProgramRunning())

{

MAINDllOBJECT->SetStatusCode("RW\_PartProgram02");

return false;

}

System::IO::StreamWriter^ tst = gcnew System::IO::StreamWriter(System::String::Concat(RWrapper::RW\_MainInterface::MYINSTANCE()->LogFolderPath,"\\ProgramLog\\ProgramDetails.txt"),true);

tst->WriteLine("Program build:");

tst->WriteLine("Program Name: " + FilePath);

tst->WriteLine("Time and Date: " + Microsoft::VisualBasic::DateAndTime::Now.ToString());

tst->WriteLine("");

tst->Close();

BuildFilePath = FilePath;

if (PPCALCOBJECT->CreateProfileLineArcFrameGrabs())

{

PartProgram\_ContinueBuild(true);

return true;

}

else

{

MAINDllOBJECT->ShowHideWaitCursor(2);

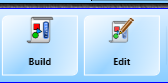
return false;

}

}

catch(Exception^ ex){ RWrapper::RW\_MainInterface::MYINSTANCE()->WriteErrorLog("WRPP0019a", ex); return false;}

}

****

**Part program Edit:**

void RWrapper::RW\_PartProgram::PartProgram\_Edit(System::String^ FilePath)

{

try

{

MAINDllOBJECT->SetStatusCode("RW\_PartProgram20");

MAINDllOBJECT->AddAction\_PPEditMode = false;

if(PPCALCOBJECT->IsPartProgramRunning())

{

MAINDllOBJECT->SetStatusCode("RW\_PartProgram02");

return;

}

if(!PPCALCOBJECT->PartprogramLoaded())

{

MAINDllOBJECT->ShowMsgBoxString("RW\_PartProgramPartProgram\_Edit01", RapidEnums::MSGBOXTYPE::MSG\_OK, RapidEnums::MSGBOXICONTYPE::MSG\_EXCLAMATION);

return;

}

if(PPCALCOBJECT->ProgramMadeUsingReferenceDot())

{

if(!PPCALCOBJECT->ReferenceDotIstakenAsHomePosition())

{

MAINDllOBJECT->ShowMsgBoxString("RW\_PartProgramPartProgram\_Edit02", RapidEnums::MSGBOXTYPE::MSG\_OK, RapidEnums::MSGBOXICONTYPE::MSG\_EXCLAMATION);

return;

}

}

if(PPCALCOBJECT->ProgramMadeUsingVblockAxis())

{

if(!PPCALCOBJECT->VblockAxisAsReference())

{

MAINDllOBJECT->ShowMsgBoxString("RW\_PartProgramPartProgram\_Edit02", RapidEnums::MSGBOXTYPE::MSG\_OK, RapidEnums::MSGBOXICONTYPE::MSG\_EXCLAMATION);

return;

}

}

bool referencedotFlag = PPCALCOBJECT->ReferenceDotIstakenAsHomePosition();

RWrapper::RW\_MainInterface::MYINSTANCE()->InternalClearAll(1);

PPCALCOBJECT->ReferenceDotIstakenAsHomePosition(referencedotFlag);

PPCALCOBJECT->Editpp((char\*)(void\*)Marshal::StringToHGlobalAnsi(FilePath).ToPointer());

UpdateProgramStepsTableForEdit();

System::IO::StreamWriter^ tst = gcnew System::IO::StreamWriter(System::String::Concat(RWrapper::RW\_MainInterface::MYINSTANCE()->LogFolderPath,"\\ProgramLog\\ProgramDetails.txt"),true);

tst->WriteLine("Program Edit:");

tst->WriteLine("Program Name: " + FilePath);

tst->WriteLine("Time and Date: " + Microsoft::VisualBasic::DateAndTime::Now.ToString());

tst->WriteLine("");

tst->Close();

RWrapper::RW\_Enum::PROGRAM\_PARAMETER PartProgramParam = RWrapper::RW\_Enum::PROGRAM\_PARAMETER(PPCALCOBJECT->ResetActionParameter());

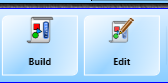
if(PartProgramParam != RWrapper::RW\_Enum::PROGRAM\_PARAMETER::PP\_NULL)

RaisePP\_ParamResetEvent(PartProgramParam);

}

catch(Exception^ ex){ RWrapper::RW\_MainInterface::MYINSTANCE()->WriteErrorLog("WRPP0011", ex); }

}

****

**Part program Run:**

void RWrapper::RW\_PartProgram::PartProgram\_Run()

{

try

{

if (PPCALCOBJECT->IsPartProgramRunning())

{

if (PPCALCOBJECT->ProgramMadeUsingTwoStepHoming() && PPCALCOBJECT->IsPartProgramRunning() && PPCALCOBJECT->FinishedManualTwoStepAlignment() || PartProgramPaused)

{// this->HandlePartprogram();

//else if (PartProgramPaused)

//{

PartProgramPaused = false;

HandlePartprogram();

}

return;

}

MAINDllOBJECT->IsPartProgramPause = false;

RWrapper::RW\_PartProgram::MYINSTANCE()->NumberOfCommandSend = 0;

this->SaveReferencePtMode = false;

//this->PointOfRotationFlag = false;

this->FixtureReferenceAngle = 0;

PartProgramPaused = false;

//PrevLightCommand = "";

for (int j = 0; j < 4; j++)

PrevTarget[j] = -10000;

PPCALCOBJECT->RefreshImageForEdgeDetection = true;

//Reset Last Lighting and CamSettings sent

MAINDllOBJECT->PrevLightCommand = "aaa";

MAINDllOBJECT->PrevCameraSetting = "bbb";

MAINDllOBJECT->SetStatusCode("RW\_PartProgram17");

if(RWrapper::RW\_DBSettings::MYINSTANCE()->GridProgramModeFlag)

{

this->CorrectRotationalError = true;

if (!RWrapper::RW\_FixtureCalibration::MYINSTANCE()->FixtureCalibValues\_Loaded)

RWrapper::RW\_FixtureCalibration::MYINSTANCE()->ReadCorrectionValues();

}

RWrapper::RW\_MainInterface::MYINSTANCE()->WriteUserLog("PartProgram\_Run", "Run PartProgram", "PartProgram Tab");

if(!PPCALCOBJECT->PartprogramLoaded())

{

RunIsnotValid::raise();

MAINDllOBJECT->ShowMsgBoxString("RW\_PartProgramPartProgram\_Run01", RapidEnums::MSGBOXTYPE::MSG\_OK, RapidEnums::MSGBOXICONTYPE::MSG\_EXCLAMATION);

return;

}

if(!PPCALCOBJECT->VblockAxisAsReference())

{

if(PPCALCOBJECT->ProgramMadeUsingVblockAxis())

{

RunIsnotValid::raise();

ShowMsgBoxString("RW\_PartProgramPartProgram\_Run04", RapidEnums::MSGBOXTYPE::MSG\_OK, RapidEnums::MSGBOXICONTYPE::MSG\_EXCLAMATION);

return;

}

}

if(PPCALCOBJECT->UseImageComparision() ||

MAINDllOBJECT->CurrentMahcineType == RapidEnums::RAPIDMACHINETYPE::ONE\_SHOT ||

MAINDllOBJECT->CurrentMahcineType == RapidEnums::RAPIDMACHINETYPE::NORMAL\_ONE\_SHOT ||

MAINDllOBJECT->CurrentMahcineType == RapidEnums::RAPIDMACHINETYPE::ONLINESYSTEM)

{

if (MAINDllOBJECT->CameraConnected())

{

if (SendCurrentStepLightCommand(PPCALCOBJECT->ppFirstgrab->getLightIntensity()) == 2)

RWrapper::RW\_CNC::MYINSTANCE()->ConvertLightValueToInt(gcnew System::String(PPCALCOBJECT->ppFirstgrab->getLightIntensity()));

std::string currCamSetting(PPCALCOBJECT->ppFirstgrab->getCamProperties());

if (MAINDllOBJECT->PrevCameraSetting != currCamSetting)

MAINDllOBJECT->SetCamProperty(PPCALCOBJECT->ppFirstgrab->getCamProperties());

MAINDllOBJECT->PrevCameraSetting = currCamSetting;

//Sleep(100); //was 1000 before

}

if(!PPCALCOBJECT->ProgramDoneForIdorOdMeasurement())

{

if (!this->SingleComponentinOneShot)

{

int Cnt = PPCALCOBJECT->GetAlltheImages();

if (Cnt == 0)

{

MAINDllOBJECT->ShowMsgBoxString("RW\_PartProgramPartProgram\_Run03", RapidEnums::MSGBOXTYPE::MSG\_OK, RapidEnums::MSGBOXICONTYPE::MSG\_EXCLAMATION);

return;

}

}

if (MAINDllOBJECT->CurrentMahcineType == RapidEnums::RAPIDMACHINETYPE::ONLINESYSTEM)

PartProgramFinished();

else

MAINDllOBJECT->ShowMsgBoxString("RW\_PartProgramPartProgram\_Run05", RapidEnums::MSGBOXTYPE::MSG\_OK, RapidEnums::MSGBOXICONTYPE::MSG\_EXCLAMATION);

return;

}

}

}

System::IO::StreamWriter^ tst = gcnew System::IO::StreamWriter(System::String::Concat(RWrapper::RW\_MainInterface::MYINSTANCE()->LogFolderPath,"\\ProgramLog\\ProgramDetails.txt"),true);

tst->WriteLine("Program Run:");

tst->WriteLine("");

tst->Close();

PPCALCOBJECT->UpdateFistFramegrab(false);

for(int i = 0; i < 4; i++)

HomePosition[i] = RWrapper::RW\_DRO::MYINSTANCE()->DROCoordinate[i];

this->ProgramMode = RWrapper::RW\_CNC::MYINSTANCE()->getCNCMode();

MAINDllOBJECT->RcadGraphicsMoveWithDRO(false);

if(PPCALCOBJECT->ProgramMadeUsingTwoReferenceDot())

{

double Point1[2] = {PPCALCOBJECT->HomePosition.getX(), PPCALCOBJECT->HomePosition.getY()}, Point2[2] = {PPCALCOBJECT->HomePosition\_2nd.getX(), PPCALCOBJECT->HomePosition\_2nd.getY()};

double Angle1 = RMATH2DOBJECT->ray\_angle(Point1, Point2);

Point1[0] = PPCALCOBJECT->PPHomePosition.getX(); Point1[1] = PPCALCOBJECT->PPHomePosition.getY();

Point2[0] = PPCALCOBJECT->PPHomePosition\_2nd.getX(); Point2[1] = PPCALCOBJECT->PPHomePosition\_2nd.getY();

double Angle2 = RMATH2DOBJECT->ray\_angle(Point1, Point2);

FixtureReferenceAngle = Angle1 - Angle2;

if (FixtureReferenceAngle < 0.000000001) FixtureReferenceAngle = 0; //If angle = 10^-9 rad, then we will have an error of 0.5um for a length of 500 mm, which is the largest we can travel in a 4030. So anything less than this shall be 0!

}

else if(PPCALCOBJECT->ProgramMadeUsingRefLine())

{

if(MAINDllOBJECT->ReferenceLineShape != NULL)

{

FixtureReferenceAngle = ((Line\*)MAINDllOBJECT->ReferenceLineShape)->Angle() - PPCALCOBJECT->ReferenceLineAngle;

if(FixtureReferenceAngle > M\_PI\_2)

FixtureReferenceAngle -= M\_PI;

else if(FixtureReferenceAngle < -M\_PI\_2)

FixtureReferenceAngle += M\_PI;

}

}

StartPartProgramRun();

}

catch(Exception^ ex){ RWrapper::RW\_MainInterface::MYINSTANCE()->WriteErrorLog("WRPP0012", ex); }

}

