```
package com.VWSA.LightUp.impl;

import com.ur.urcap.api.contribution.ProgramNodeContribution;
import com.ur.urcap.api.contribution.program.ProgramAPIProvider;
import com.ur.urcap.api.domain.UserInterfaceAPI;
import com.ur.urcap.api.domain.data.DataModel;
import com.ur.urcap.api.domain.script.ScriptWriter;
import com.ur.urcap.api.domain.undoredo.UndoRedoManager;
import com.ur.urcap.api.domain.undoredo.UndoableChanges;
import com.ur.urcap.api.domain.value.Pose;
import com.ur.urcap.api.domain.value.ValueFactoryProvider;
import com.ur.urcap.api.domain.userinteraction.keyboard.KeyboardInputCallback;
import com.ur.urcap.api.domain.userinteraction.keyboard.KeyboardInputFactory;
import com.ur.urcap.api.domain.userinteraction.keyboard.KeyboardTextInput;
import com.ur.urcap.api.domain.userinteraction.robot.movement.MovementCompleteEvent;
import com.ur.urcap.api.domain.userinteraction.robot.movement.MovementErrorEvent;
import com.ur.urcap.api.domain.userinteraction.robot.movement.RobotMovement;
import com.ur.urcap.api.domain.userinteraction.robot.movement.RobotMovementCallback;
import com.ur.urcap.api.domain.value.jointposition.JointPositions;
import com.ur.urcap.api.domain.value.simple.Angle;
import com.ur.urcap.api.domain.value.simple.Length;
import com.ur.urcap.api.domain.userinteraction.RobotPositionCallback;

public class LightUpProgramNodeContribution implements ProgramNodeContribution{
public static Integer Test = 1;
            private final ProgramAPIProvider apiProvider;
            private final LightUpProgramNodeView view;
            private final DataModel model;
            private final UndoRedoManager undoRedoManager;
            private final KeyboardInputFactory keyboardInputFactory;
            private static final String PosX_KEY = "PosX_Key";
            private static final String PosY_KEY = "PosY_Key";
            private static final String PosZ_KEY = "PosZ_Key";
            private static final String RotRX_Key = "RotRX_Key";
            private static final String RotRY_KEY = "RotRY_Key";
            private static final String RotRZ_KEY = "RotRZ_Key";
            private static final String Vel_KEY = "Vel_Key";
            private static final String Acc_KEY = "Acc_Key";
            private static final String Blend_KEY = "Blend_Key";
            private static final String X_Offs_KEY = "X_Offs_KEY";
            private static final String Y_Offs_KEY = "Y_Offs_KEY";
            private static final String Z_Offs_KEY = "Z_Offs_KEY";
            private final RobotMovement robotMovement;
            private static final String Position_KEY = "Position_Key";
            private static final String Step_Position_KEY = "Step_Position_Key";

//*************************************
//MoveL Node Contribution
//*************************************
            public LightUpProgramNodeContribution(ProgramAPIProvider apiProvider, LightUpProgramNodeView view, DataModel model) {
                        this.apiProvider = apiProvider;
                        this.view = view;
                        this.model = model;
                        this.keyboardInputFactory = apiProvider.getUserInterfaceAPI().getUserInteraction().getKeyboardInputFactory();
                        this.undoRedoManager = this.apiProvider.getProgramAPI().getUndoRedoManager();
                        robotMovement = apiProvider.getUserInterfaceAPI().getUserInteraction().getRobotMovement();
            }
//*************************************
//Touch Up Position Minus Offsets
//*************************************
            public void TouchUp(final Integer output) {
                        UserInterfaceAPI uiapi = apiProvider.getUserInterfaceAPI();
                        uiapi.getUserInteraction().getUserDefinedRobotPosition(new RobotPositionCallback() {
                                    @Override
                                    public void onOk(Pose pTouchUpPose, JointPositions jointpositions) {
                                                model.set(PosX_KEY, pTouchUpPose.getPosition().getX(Length.Unit.M)-getX_Offs());
                                                model.set(PosY_KEY, pTouchUpPose.getPosition().getY(Length.Unit.M)-getY_Offs());
                                                model.set(PosZ_KEY, pTouchUpPose.getPosition().getZ(Length.Unit.M)-getZ_Offs());
                                                model.set(RotRX_Key, pTouchUpPose.getRotation().getRX(Angle.Unit.RAD));
                                                model.set(RotRY_KEY, pTouchUpPose.getRotation().getRY(Angle.Unit.RAD));
                                                model.set(RotRZ_KEY, pTouchUpPose.getRotation().getRZ(Angle.Unit.RAD));
                                                ValueFactoryProvider valueFactoryProvider = apiProvider.getProgramAPI().getValueFactoryProvider();
                                                model.set(Position_KEY, valueFactoryProvider.getPoseFactory().createPose(getPos_X(), getPos_Y(),
getPos_Z(), getRot_RX(), getRot_RY(), getRot_RZ(), Length.Unit.M, Angle.Unit.RAD));
                                    }
                        });
            }
//*************************************
//Get Position X For Model
//*************************************
            private Double getPos_X() {
                        return model.get(PosX_KEY, 0.0);
            }
//*************************************
//Get Position Y For Model
//*************************************
            private Double getPos_Y() {
                        return model.get(PosY_KEY, 0.0);
            }
//*************************************
//Get Position Z For Model
//*************************************
            private Double getPos_Z() {
                        return model.get(PosZ_KEY, 0.0);
            }
//*************************************
//Get Rotation X For Model
//*************************************
            private Double getRot_RX() {
                        return model.get(RotRX_Key, 0.0);
            }
//*************************************
//Get Rotation Y For Model
//*************************************
            private Double getRot_RY() {
                        return model.get(RotRY_KEY, 0.0);
            }
//*************************************
//Get Rotation Z For Model
//*************************************
            private Double getRot_RZ() {
                        return model.get(RotRZ_KEY, 0.0);
            }
//*************************************
//Get Acceleration For Model
//*************************************
            private Double getAcc() {
                        return model.get(Acc_KEY, 0.0);
            }
//*************************************
//Get Velocity For Model
//*************************************
            private Double getVel() {
```

```
			return model.get(Vel_KEY, 0.0);
		}
	//*************************************
	//Get Blend For Model
	//*************************************
		private Double getBlend() {
			return model.get(Blend_KEY, 0.0);
		}
	//*************************************
	//Get X Offset For Model
	//*************************************
		private Double getX_Offs() {
			return model.get(X_Offs_KEY, 0.0)/1000;
		}
	//*************************************
	//Get Y Offset For Model
	//*************************************
		private Double getY_Offs() {
			return model.get(Y_Offs_KEY, 0.0)/1000;
		}
	//*************************************
	//Get Z Offset For Model
	//*************************************
		private Double getZ_Offs() {
			return model.get(Z_Offs_KEY, 0.0)/1000;
		}
	//*************************************
	//Acceleration Keyboard Input
	//*************************************
		public KeyboardTextInput getKeyboardForAcc() {
			KeyboardTextInput keyboard = keyboardInputFactory.createStringKeyboardInput();
			keyboard.setInitialValue(model.get(Acc_KEY, ""));
			return keyboard;
		}
		public KeyboardInputCallback<String> getCallbackForAcc() {
			return new KeyboardInputCallback<String>() {
				@Override
				public void onOk(String value) {
					view.setAcc(value);
					model.set(Acc_KEY, value);
				}
			};
		}
	//*************************************
	//Velocity Keyboard Input
	//*************************************
		public KeyboardTextInput getKeyboardForVel() {
			KeyboardTextInput keyboard = keyboardInputFactory.createStringKeyboardInput();
			keyboard.setInitialValue(model.get(Vel_KEY, ""));
			return keyboard;
		}
		public KeyboardInputCallback<String> getCallbackForVel() {
			return new KeyboardInputCallback<String>() {
				@Override
				public void onOk(String value) {
					view.setVel(value);
					model.set(Vel_KEY, value);
				}
			};
		}
	//*************************************
	//Blend Keyboard Input
	//*************************************
		public KeyboardTextInput getKeyboardForBlend() {
			KeyboardTextInput keyboard = keyboardInputFactory.createStringKeyboardInput();
			keyboard.setInitialValue(model.get(Blend_KEY, ""));
			return keyboard;
		}
		public KeyboardInputCallback<String> getCallbackForBlend() {
			return new KeyboardInputCallback<String>() {
				@Override
				public void onOk(String value) {
					view.setBlend(value);
					model.set(Blend_KEY, value);
				}
			};
		}
	//*************************************
	//X Offset Keyboard Input
	//*************************************
		public KeyboardTextInput getKeyboardForX_Offs() {
			KeyboardTextInput keyboard = keyboardInputFactory.createStringKeyboardInput();
			keyboard.setInitialValue(model.get(X_Offs_KEY, ""));
			return keyboard;
		}
		public KeyboardInputCallback<String> getCallbackForX_Offs() {
			return new KeyboardInputCallback<String>() {
				@Override
				public void onOk(String value) {
					view.setX_Offs(value);
					model.set(X_Offs_KEY, value);
				}
			};
		}
	//*************************************
	//Y Offset Keyboard Input
	//*************************************
		public KeyboardTextInput getKeyboardForY_Offs() {
			KeyboardTextInput keyboard = keyboardInputFactory.createStringKeyboardInput();
			keyboard.setInitialValue(model.get(Y_Offs_KEY, ""));
			return keyboard;
		}
		public KeyboardInputCallback<String> getCallbackForY_Offs() {
			return new KeyboardInputCallback<String>() {
				@Override
				public void onOk(String value) {
					view.setY_Offs(value);
					model.set(Y_Offs_KEY, value);
				}
			};
		}
	//*************************************
	//Z Offset Keyboard Input
	//*************************************
		public KeyboardTextInput getKeyboardForZ_Offs() {
			KeyboardTextInput keyboard = keyboardInputFactory.createStringKeyboardInput();
			keyboard.setInitialValue(model.get(Z_Offs_KEY, ""));
			return keyboard;
		}
		public KeyboardInputCallback<String> getCallbackForZ_Offs() {
			return new KeyboardInputCallback<String>() {
				@Override
				public void onOk(String value) {
```

```
                                                view.setZ_Offs(value);
                                                model.set(Z_Offs_KEY, value);
                                    }
                        };
            }
//*************************************
//Move Robot To Offset Position
//*************************************
            public void moveRobot() {
                        Offs();
                        Pose OffsetPose = model.get(Step_Position_KEY, (Pose) null);
                        if (OffsetPose != null) {
                                    robotMovement.requestUserToMoveRobot(OffsetPose, new RobotMovementCallback() {
                                                @Override
                                                public void onComplete(MovementCompleteEvent event) {

                                                }
                                                @Override
                                                public void onError(MovementErrorEvent event) {

                                                }
                                    });
                        }
            }
            public void Offs() {
                        undoRedoManager.recordChanges(new UndoableChanges() {
                                    @Override
                                    public void executeChanges() {
                                                ValueFactoryProvider valueFactoryProvider = apiProvider.getProgramAPI().getValueFactoryProvider();
                                                model.set(Step_Position_KEY, valueFactoryProvider.getPoseFactory().createPose(getPos_X()+getX_Offs(),
getPos_Y()+getY_Offs(), getPos_Z()+getZ_Offs(), getRot_RX(), getRot_RY(), getRot_RZ(), Length.Unit.M, Angle.Unit.RAD));
                                    }
                        });
            }
//*************************************
//Open View
//*************************************
            @Override
            public void openView() {
                        getPos_X();
                        getPos_Y();
                        getPos_Z();
                        getRot_RX();
                        getRot_RY();
                        getRot_RZ();
                        getAcc();
                        getVel();
                        getBlend();
                        getX_Offs();
                        getY_Offs();
                        getZ_Offs();
                        if (getAcc() == 0) {
                                    model.set(Acc_KEY, 1200);
        }
                        if (getVel() == 0) {
                                    model.set(Vel_KEY, 300);
        }
                        if (getBlend() == 0) {
                                    model.set(Blend_KEY, 0);
        }
                        getAcc();
                        getVel();
                        getBlend();
                        view.setAcc(model.get(Acc_KEY, ""));
                        view.setVel(model.get(Vel_KEY, ""));
                        view.setBlend(model.get(Blend_KEY, ""));
                        model.get(Position_KEY, (Pose) null);
            }
//*************************************
//Close View
//*************************************
            @Override
            public void closeView() {
            }
//*************************************
//Get Title
//*************************************
            @Override
            public String getTitle() {
                        return "VW_MoveL";
            }
//*************************************
//Is Defined
//*************************************
            @Override
            public boolean isDefined() {
                        return true;
            }
//*************************************
//Generate Script
//*************************************
            @Override
            public void generateScript(ScriptWriter writer) {
                        writer.assign("CameraOffset", "p["+getX_Offs()+", "+getY_Offs()+", "+getZ_Offs()+", 0.0, 0.0, 0.0]");
                        writer.appendLine("Position = "+model.get(Position_KEY, (Pose) null));
                        writer.appendLine("movel(pose_trans(CameraOffset, Position), a="+(getAcc()/1000)+", v="+(getVel()/1000)+",
r="+(getBlend()/1000)+")");
            }
}
```