```python
import maya.cmds as cmds

class ReparentTool(object):

    def __init__(self):
        self.window_id = "reparentToolWindow"
        self.create_ui()

    def create_ui(self):
        if cmds.window(self.window_id, exists=True):
            cmds.deleteUI(self.window_id)

        self.window = cmds.window(
            self.window_id,
            title="Reparent Tool",
            widthHeight=(500, 160),
            sizeable=True
        )

        main_layout = cmds.columnLayout(adjustableColumn=True, rowSpacing=10)

        # Children Row
        cmds.rowLayout(
            numberOfColumns=3,
            adjustableColumn=2,
            columnWidth3=(80, 300, 100),
            columnAttach=[(1, 'both', 0), (2, 'both', 5), (3, 'both', 0)],
            parent=main_layout
            )
        cmds.text(label="Children", align="left")
        self.children_txt = cmds.textField()
        cmds.button(label="Get Selection", command=self.get_children_selection)

        # Parent Row
        cmds.rowLayout(
            numberOfColumns=3,
```

```
        adjustableColumn=2,
        columnWidth3=(80, 300, 100),
        columnAttach=[(1, 'both', 0), (2, 'both', 5), (3, 'both', 0)],
        parent=main_layout
        )
    cmds.text(label="Parent", align="left")
    self.parent_txt = cmds.textField()
    cmds.button(label="Get Selection", command=self.get_parent_selection)

    # Buttons
    cmds.setParent(main_layout)
    cmds.separator()
    cmds.button(label="Create Relative Locator(s)", height=35, command=self.on_create_relative_locators)
    cmds.button(label="Reparent Child(ren) to Parent", height=35, command=self.on_reparent_children)

    cmds.showWindow(self.window)

def get_children_selection(self, *args):
    selection = cmds.ls(sl=True, long=True)
    if selection:
        cmds.textField(self.children_txt, edit=True, text="; ".join(selection))
    else:
        cmds.textField(self.children_txt, edit=True, text="")

def get_parent_selection(self, *args):
    selection = cmds.ls(sl=True, long=True)
    if selection:
        cmds.textField(self.parent_txt, edit=True, text=selection[0])
    else:
        cmds.textField(self.parent_txt, edit=True, text="")

def get_inputs(self):
    children_raw = cmds.textField(self.children_txt, q=True, text=True).split("; ")
    children = [c.strip() for c in children_raw if cmds.objExists(c.strip())]

    parent_raw = cmds.textField(self.parent_txt, q=True, text=True).strip()
```

```
    parent = parent_raw if cmds.objExists(parent_raw) else None

    return children, parent

def get_time_range(self):
    start = cmds.playbackOptions(q=True, minTime=True)
    end = cmds.playbackOptions(q=True, maxTime=True)
    return start, end

def on_create_relative_locators(self, *args):
    children, parent = self.get_inputs()

    if not children or not parent:
        cmds.warning("WARNING! Select both child(ren) object and a parent object.")
        return

    start_frame, end_frame = self.get_time_range()
    locators_created = []

    for child in children:
        loc = cmds.spaceLocator(name="{}_relative_LOC".format(child.split('|')[-1]))[0]
        cmds.parent(loc, parent)
        cmds.matchTransform(loc, child, pos=True, rot=True)
        constraint = cmds.parentConstraint(child, loc, maintainOffset=True)[0]

        cmds.bakeResults(
            loc,
            time=(start_frame, end_frame),
            simulation=True,
            disableImplicitControl=True,
            controlPoints=False,
            shape=False
        )

        if cmds.objExists(constraint):
            cmds.delete(constraint)
```

```
        locators_created.append(loc)

    cmds.select(locators_created)
    print("Successfully created relative locator(s): {}".format(locators_created))


def on_reparent_children(self, *args):
    children, parent = self.get_inputs()

    if not children or not parent:
        cmds.warning("Please ensure both valid Children and a Parent are specified.")
        return

    start_frame, end_frame = self.get_time_range()

    cmds.undoInfo(openChunk=True)
    try:
        for child in children:

            loc = cmds.spaceLocator(name="{}_world_LOC".format(child.split('|')[-1]))[0]

            temp_const1 = cmds.parentConstraint(child, loc, maintainOffset=False)[0]

            cmds.bakeResults(
                loc,
                time=(start_frame, end_frame),
                simulation=True,
                disableImplicitControl=True,
                controlPoints=False,
                shape=False
            )
            cmds.delete(temp_const1)

            cmds.cutKey(child, attribute=['tx', 'ty', 'tz', 'rx', 'ry', 'rz'], clear=True)
```

```
            reparented_result = cmds.parent(child, parent)

            child_new_long_name = cmds.ls(reparented_result[0], long=True)[0]

            temp_const2 = cmds.parentConstraint(loc, child_new_long_name, maintainOffset=False)[0]

            cmds.bakeResults(
                child_new_long_name,
                time=(start_frame, end_frame),
                simulation=True,
                disableImplicitControl=True,
                controlPoints=False,
                shape=False
            )

            if cmds.objExists(temp_const2):
                cmds.delete(temp_const2)
            cmds.delete(loc)

        print("Successfully reparented child animations and compensated relative TRS channels.")

    finally:
        cmds.undoInfo(closeChunk=True)

ReparentTool()
```