```cpp
using namespace juce;

namespace evt {
enum class Type {
  ParameterEditorShowUp,
  ParameterEditorAssignParam,
  ParameterEditorVanish,
  // ... more
};
}

namespace param {
using PID = int;
}

namespace gui {

struct KeyPress;
struct Utils;

struct TextEditor : public juce::Component {

  // u, emptyString
  TextEditor(Utils&, String const& = "");
  void setText(String const&);
  virtual void setActive(bool e);
  void addEvt(evt::Evt const&);
  // ... more

  String txt;
  std::function<void()> onEnter;
};

struct ParameterEditor : TextEditor {
  ParameterEditor(Utils& u)
      : TextEditor(u)
      , pIDs()
  {
    onEnter = [&]() {
      setActive(false);
      for (auto const pID : pIDs) {
        auto& param = u.getParam(pID);
        auto const valDenormTxt = txt;
        auto const valDenorm = param.getValForTextDenorm(valDenormTxt);
        auto const valLegal = param.range.snapToLegalValue(valDenorm);
        auto const valNorm = param.range.convertTo0to1(valLegal);
        param.setValueWithGesture(valNorm);
      }
    };

    addEvt([&](evt::Type t, void const* stuff) {
      if (t == evt::Type::ParameterEditorShowUp) {
        auto const pluginScreenBounds = u.pluginTop.getScreenBounds();
        auto const screenBoundsParent = getParentComponent()->getScreenBounds();
        auto const screenBounds = getScreenBounds();
        auto const knobScreenBounds = *static_cast<Bounds const *>(stuff);
        auto const x = knobScreenBounds.getX() - pluginScreenBounds.getX();
        auto const y = knobScreenBounds.getY() - pluginScreenBounds.getY();
        setTopLeftPosition(x, y);
        setActive(true);
      } else if (t == evt::Type::ParameterEditorAssignParam) {
        pIDs = *static_cast<std::vector<param::PID> const*>(stuff);
        auto const& param = u.getParam(pIDs[0]);
        setText(param.getCurrentValueAsText());
        repaint();
      } else if (t == evt::Type::ParameterEditorVanish) {
        setActive(false);
      }
    });
  }

  void setActive(bool e) override
  {
    if (e)
      setVisible(e);
    TextEditor::setActive(e);
    if (!e)
      setVisible(e);
  }

  private:
  std::vector<param::PID> pIDs;
};
} // namespace gui

```