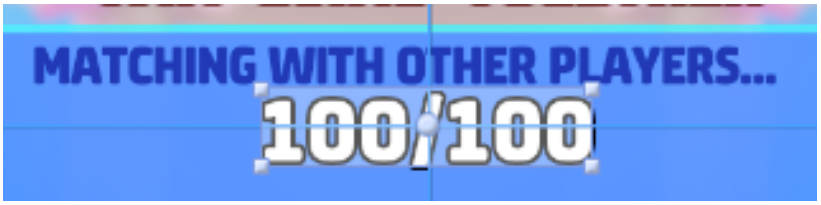
Q3`25-Slots-修复引擎 bug- 滚动字体更新透明度、颜色异常-程序

问题表现：

- 时间线中存在 Opacity、color 关键帧，导致初始化时已经存在但未实际显示的 charNode 在需要显示时透明度、颜色异常的情况



滚动更新后尾部 char 丢失：

- jackpot 滚动数字，时间线中存在 Opacity、color 关键帧，滚动过程中会出现某位数字颜色、透明度显示异常的情况：

核心问题：

- 节点属性正常（visible、color、opacity 等），但未触发渲染接口重绘，需要更新对应的 dirtyFlag；
- CCLableBMFont 默认启用了级联颜色_cascadeColorEnabled、级联透明度 _cascadeOpacityEnabled，依赖级联传递更新 dirtyFlag

解决方案：

方法一：更新 text 时更新对应 dirtyFlag

```
if (cc.sys.isObjectValid(node) && node._renderCmd) {
    node._renderCmd.setDirtyFlag(cc.Node._dirtyFlags.opacityDirty);
    node._renderCmd.setDirtyFlag(cc.Node._dirtyFlags.colorDirty);
}
```

方法二：直接调用公开接口，内部刷新 dirty 标记位 (✅)

nodeHelper.js

```
setNodeText: function(node, text, isScaleY){
    if(node && cc.sys.isObjectValid(node)){
        if(cc.isUndefined(text) || null === text){
            text = "null";
        }else{
            text = text.toString();
        }
        node.setString(text);

        //#region 解决引擎 bug:  时间线中存在 Opacity、color 关键帧，
        //导致初始化时已经存在但未实际显示的 charNode 在需要显示时透明度、颜色异常的情况
        //有效，内部调用了 setDirtyFlag(cc.Node._dirtyFlags.opacityDirty)
        node.setOpacity(node.getOpacity());
        //有效，内部调用了 setDirtyFlag(cc.Node._dirtyFlags.colorDirty)
        node.setColor(node.getColor());
        //#endregion

        if(node.__oriLabelScale && node.__oriLabelWidth && game && game.util){
            game.util.scaleCCLabelBMFontWithOriScale(node, node.__oriLabelWidth, node.__oriLabelScale, isScaleY? node.__oriLabelScaleY:undefined);
        }
    }
}
```

方法三：保证 CCB 中初始 char 个数小于后续可能出现的 char 个数

- 使得每多显示一个 charNode 都是通过 addchild 新增的；
- 例如：从 0/100 递增变化为 100/100，需要保证 CCB 中设置的初始字符串个数小于 5，可以是 0、0/0、0/10 等；

- 原因：CCNode 的 addChild 内部接口 addchildHelper 中，若本节点启用了级联颜色_cascadeColorEnabled、级联透明度_cascadeOpacityEnabled，会立即触发一次本节点对应标记位刷新；

其他方案：

- `fontnode.setNodeDirty(true);` //无效 ❌

  因为这里的 dirtyFlag 是 cc.Node._dirtyFlags.transformDirty

- `fontnode.setVisible(true);` //无效 ❌

  因为 CCNode 内部有 visible 是否变化的判定，且最终触发的是 `setDirtyFlag(cc.Node._dirtyFlags.transformDirty);`

- `fontnode.getChildren()[6].setVisible(true);` //有效但不建议 ⚠️

  原因一：不方便，需要直接操作 charNode；

  原因二：此方法适用于 CCSprite，是因为 CCSprite 的 setVisible 中补充了 this._renderCmd.setDirtyRecursively(true) 调用，
  但在 CanvasRender、WebGLRender、OpenGlRender 中该方法存在差异，比如 CanvasRender 会将 _renderCmd.setDirtyRecursively 重写为空函数；

- `fontnode.getChildren()[6].dirty = true;` //无效 ❌

  CCSprite 的 dirty 属性不同于 dirtyFlag 并不会直接触发渲染接口重绘