













```
// Make track one
std::vector<std::unique_ptr<Node>> trackOneClipNodes;
trackOneClipNodes.push_back (std::make_unique<SinNode> (220.0f, 1));
trackOneClipNodes.push_back (std::make_unique<SinNode> (220.0f, 1));

auto trackOneNode = std::make_unique<SummingNode> (std::move (trackOneClipNodes));

// Make track two
auto trackTwoClipNode = std::make_unique<SinNode> (220.0f, 1);
float clipGain = 1.0f;
auto trackTwoNode = std::make_unique<GainNode> (std::move (trackTwoClipNode),
                                                [clipGain] { return clipGain; });

// Make main output node
std::vector<std::unique_ptr<Node>> trackNodes;
trackNodes.push_back (std::move (trackOneNode));
trackNodes.push_back (std::move (trackTwoNode));
auto mainOutput = std::make_unique<SummingNode> (std::move (trackNodes));

// Play mainOutput!
```

**Main output**



**Sin**



**Sin**

**Track**



**Sin**

**Track**



std::vector<std::unique\_ptr<

trackOneC1ipNodes.push\_back

stod





uniquu

e

s





uniquu

```
>(std::move(trackOneClipNodes));
```





trackNodes.push\_back(







uniquu

std::move(trackTwoC1pNode),



uniquely



E

S







std::vector<std::unique\_ptr<Node>> track



uniquely

stod. ■ ■

auto track two clip

```
>(std::move(trackNodes));
```

stod



es

=

=

s

tc

d



en>>trackoneclip



~~/n/~~Marketplace

<GainNode>

/// Make main output node

autostrade

ktwoNode =

~~/~~Marketplace two

autostrade

auto main input =

std::move(trackedNode)



`<sinNode>(220_of, 1)`

Summing Node

///PlaymainOutput!

```
std::move(trackTwoNode));
```

make

float clipGain = 1.0f;

```
[c\ipGain]{return c\ipGain;}
```

Summing Node





make

e<sinNode>(220.f, 1)

make

**No**

**Nod**

**No**

make



**nod**













~~/n/~~Marketplace



<GainNode>

/// Make main output node

autostrade

ktwoNode =

~~/~~Marketplace two

autostrade

auto main input =

std::move(trackedNode)

`<sinNode>(220.f, 1)`



# Summing Node

///PlaymainOutput!

```
std::move(trackTwoNode));
```

make

```
float clipGain = 1.0f;
```

```
[c\ipGain]{return c\ipGain;}
```

# Summing Node





make

`<sinNode>(220.f, 1)`

make



nod



nod

**Not**



make

make