Intermediate Programming

Day 18

Outline

- Exercise 6-2
- Midterm project

Exercise 6-2 (part 2)

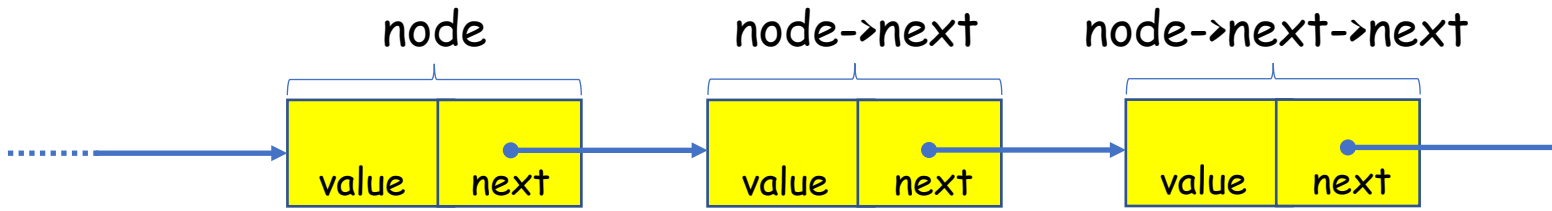
Implement the **remove_after** and **remove_front** functions.

```
char remove_after( Node *node )
{




}
```

Exercise 6-2 (part 2)

Implement the **remove_after** and **remove_front** functions.

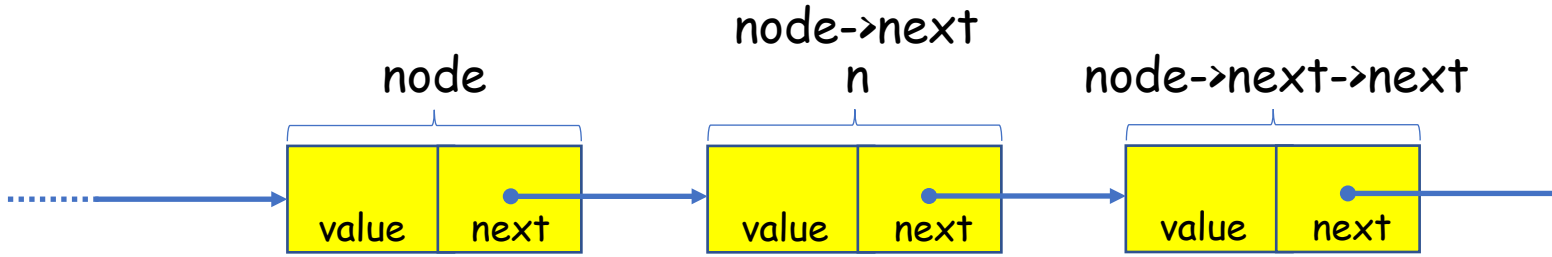
```
char remove_after( Node *node )
{




}
```



Exercise 6-2 (part 2)

Implement the **remove_after** and **remove_front** functions.

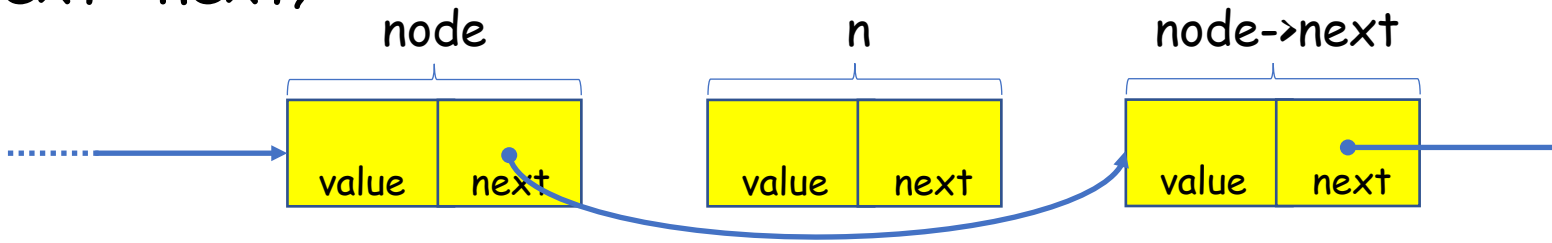
```
char remove_after( Node *node )
{
    Node *n = node->next;




}
```



Exercise 6-2 (part 2)

Implement the **remove_after** and **remove_front** functions.

```
char remove_after( Node *node )
{
    Node *n = node->next;
    if( !n ) return '?';
    char data = n->data;
    node->next = node->next->next;


}
```



Exercise 6-2 (part 2)

Implement the **remove_after** and **remove_front** functions.

```
char remove_after( Node *node )
{
    Node *n = node->next;
    if( !n ) return '?';
    char data = n->data;
    node->next = node->next->next;
    free( n );

}
```

Exercise 6-2 (part 2)

Implement the **remove_after** and **remove_front** functions.

```
char remove_after( Node *node )
{
    Node *n = node->next;
    if( !n ) return '?';
    char data = n->data;
    node->next = node->next->next;
    free( n );
    return data;
}
```

Exercise 6-2 (part 2)

Implement the **remove_after** and **remove_front** functions.

```
char remove_front( Node **list_ptr )
{




}
```

Exercise 6-2 (part 2)

Implement the **remove_after** and **remove_front** functions.

```
char remove_front( Node **list_ptr )
{




}
```

Exercise 6-2 (part 2)

Implement the **remove_after** and **remove_front** functions.

```
char remove_front( Node **list_ptr )
{
    Node* n = (*list_ptr);




}
```

Exercise 6-2 (part 2)

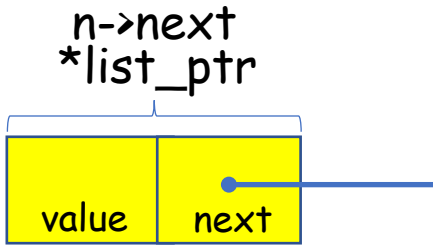
Implement the **remove_after** and **remove_front** functions.

```
char remove_front( Node **list_ptr )
{
    Node* n = (*list_ptr);
    if( !n ) return '?';
    char data = n->data;
    *list_ptr = n->next;



}
```

Exercise 6-2 (part 2)

Implement the **remove_after** and **remove_front** functions.

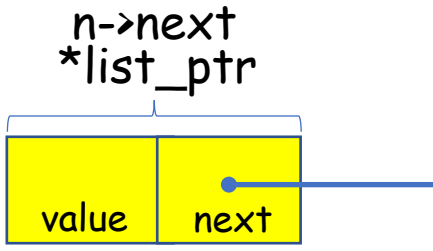
```
char remove_front( Node **list_ptr )
{
    Node* n = (*list_ptr);
    if( !n ) return '?';
    char data = n->data;
    *list_ptr = n->next;
    free( n );

}
```



Exercise 6-2 (part 2)

Implement the **remove_after** and **remove_front** functions.

```
char remove_front( Node **list_ptr )
{
    Node* n = (*list_ptr);
    if( !n ) return '?';
    char data = n->data;
    *list_ptr = n->next;
    free( n );
    return data;
}
```



Exercise 6-2 (part 4)

Implement the **remove_all** function.

```
void remove_all( Node **list_ptr , char val )
{
    while( (*list_ptr)->data==val ) remove_front( list_ptr );
    for( Node *n=*list_ptr ; n ; n=n->next )
        if( n->next && n->next->data==val )
            remove_after( n );
}
```

Exercise 6-2 (part 4)

Implement the **remove_all** function.

```
void remove_all( Node **list_ptr , char val )
{
    while( (*list_ptr)->data==val ) remove_front( list_ptr );
    for( Node *n=*list_ptr ; n ; n=n->next )
            while( n->next && n->next->data==val )
                    remove_after( n );
}
```

Exercise 6-2 (part 6)

Implement the **insert** function.

```
Node *insert( Node **list_ptr , char val )
{
    if( !*list_ptr )
    {
            *list_ptr  = create_node( val );
            return *list_ptr;
    }
    else if( val<(*list_ptr)->data )
    {
            add_front( list_ptr , val );
            return *list_ptr;
    }
    else
    {
            Node *n;
            for( n=*list_ptr ; n->next!=NULL && val>=n->next->data ; n=n->next ) ;
            add_after( n , val );
            return n->next;
    }
}
```

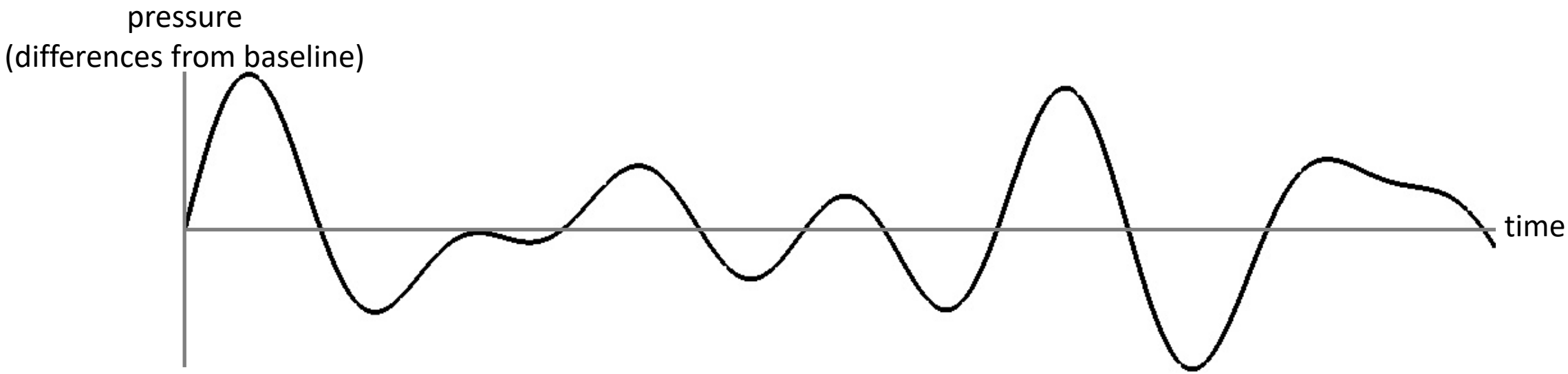
Outline

- Exercise 6-2
- Midterm project

Midterm project

Audio is represented as an oscillating function of time (seconds).

- This describes the disturbance in the ambient pressure

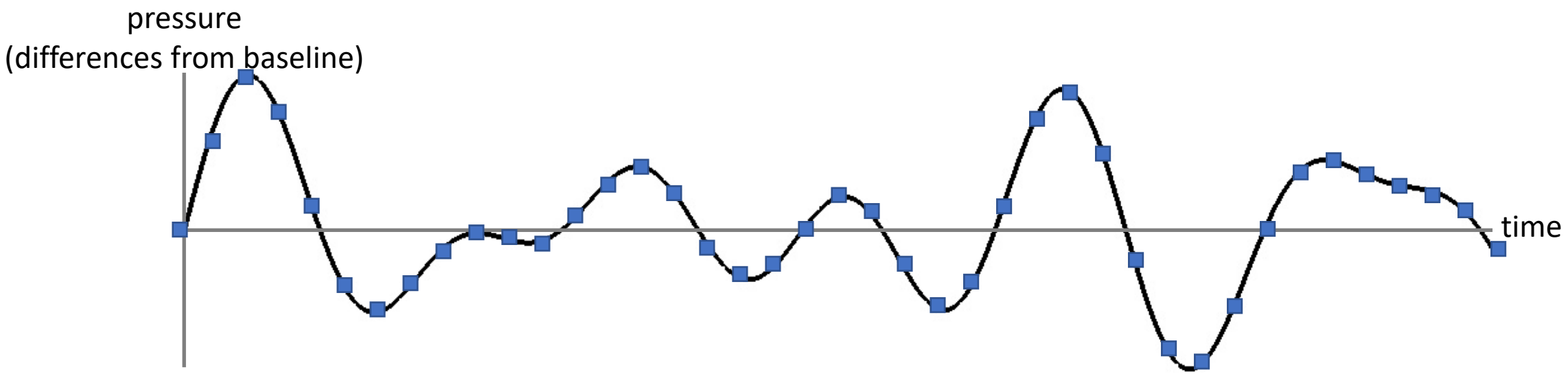


Midterm project

Audio is represented as an oscillating function of time (seconds).

To represent it on a computer, we represent it by discrete samples.

- The number of samples/second is Hertz
- In the assignment this is fixed at 44.1KHz



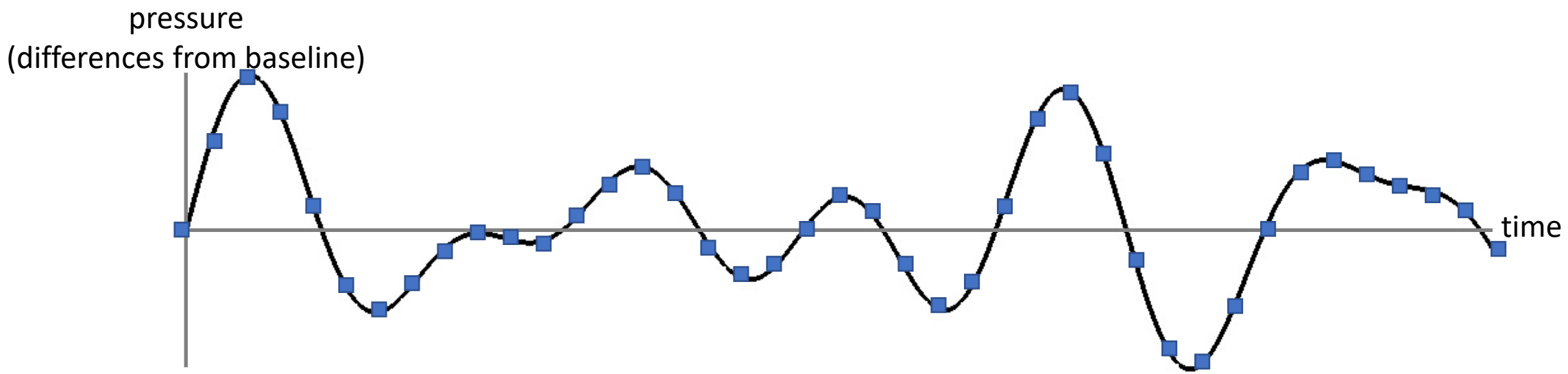
Midterm project

Audio is represented as an oscillating function of time (seconds).

To represent it on a computer, we represent it by discrete samples.

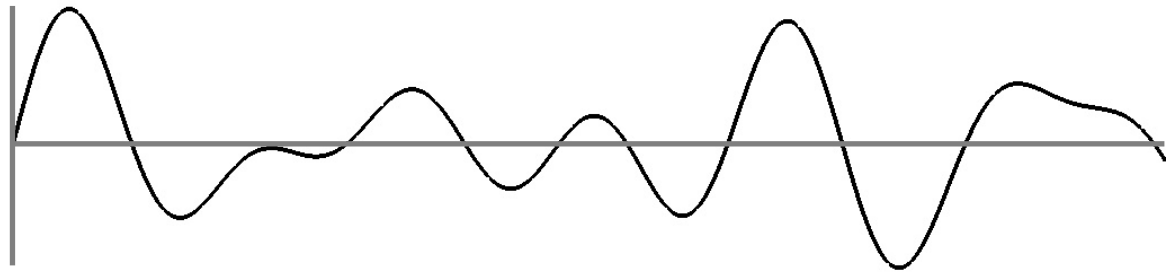
When we represent an audio signal in stereo, we represent (and sample) two signals, one for the left ear and one for the right.

- We need twice as many samples



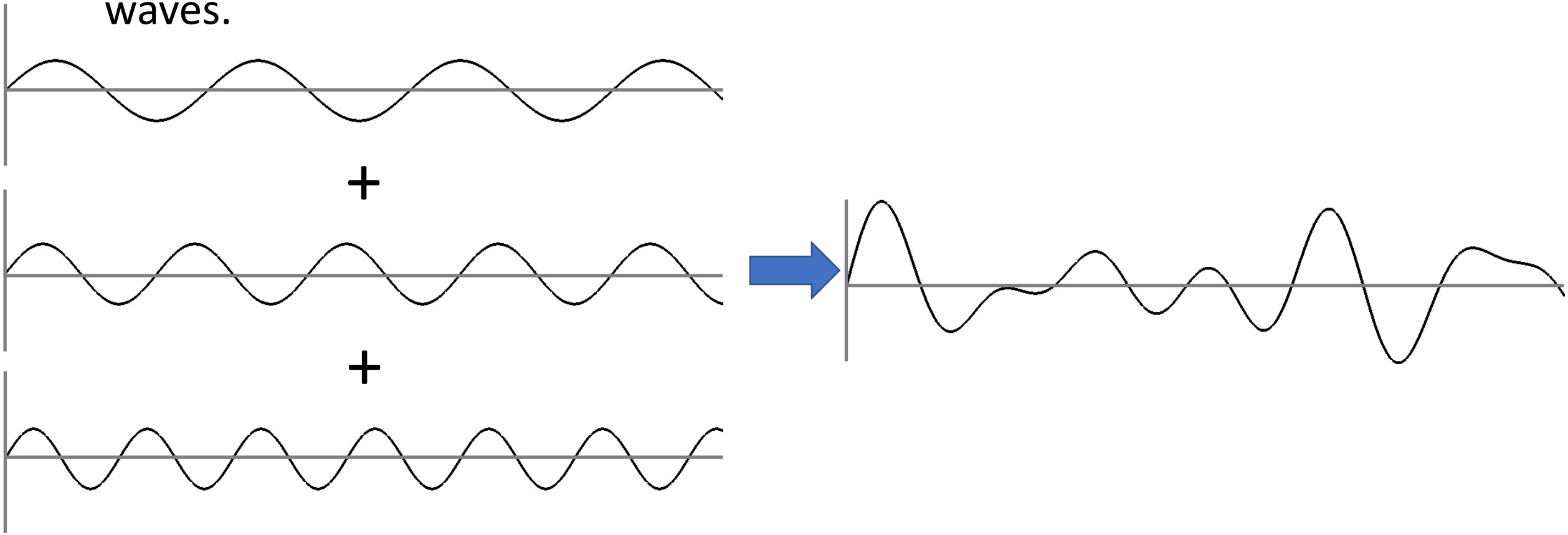
Midterm project

As the signal is (locally) oscillatory,



Midterm project

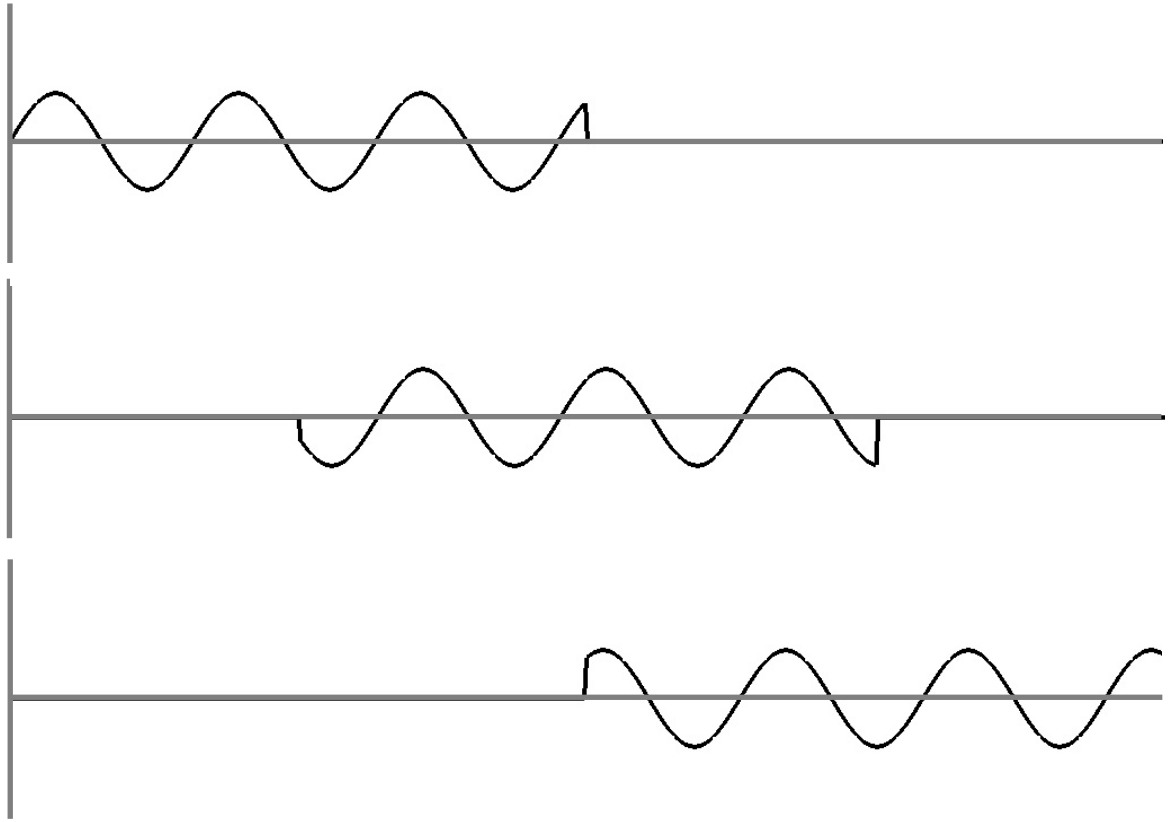
As the signal is (locally) oscillatory, we represent it as a sum of different waves.



Midterm project

The individual waves are represented in terms of their:

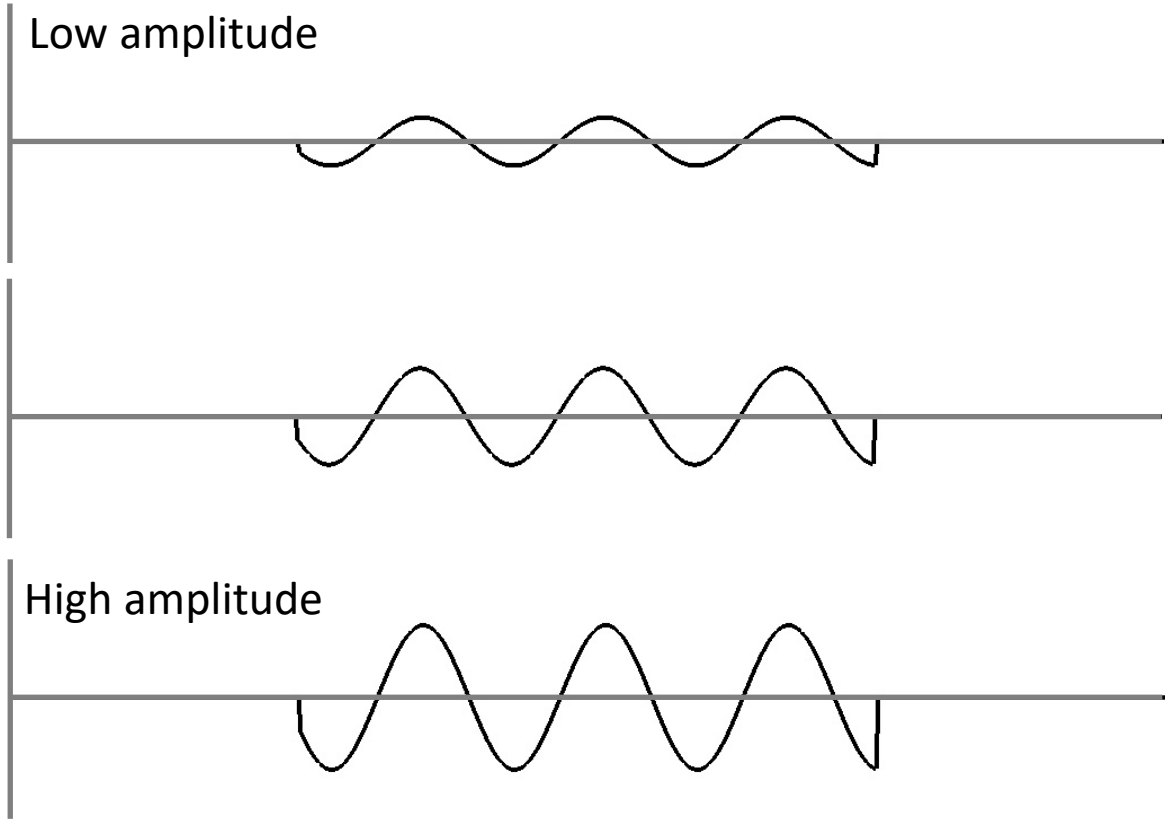
- **Temporal span**
  When is the sound played?



Midterm project

The individual waves are represented in terms of their:

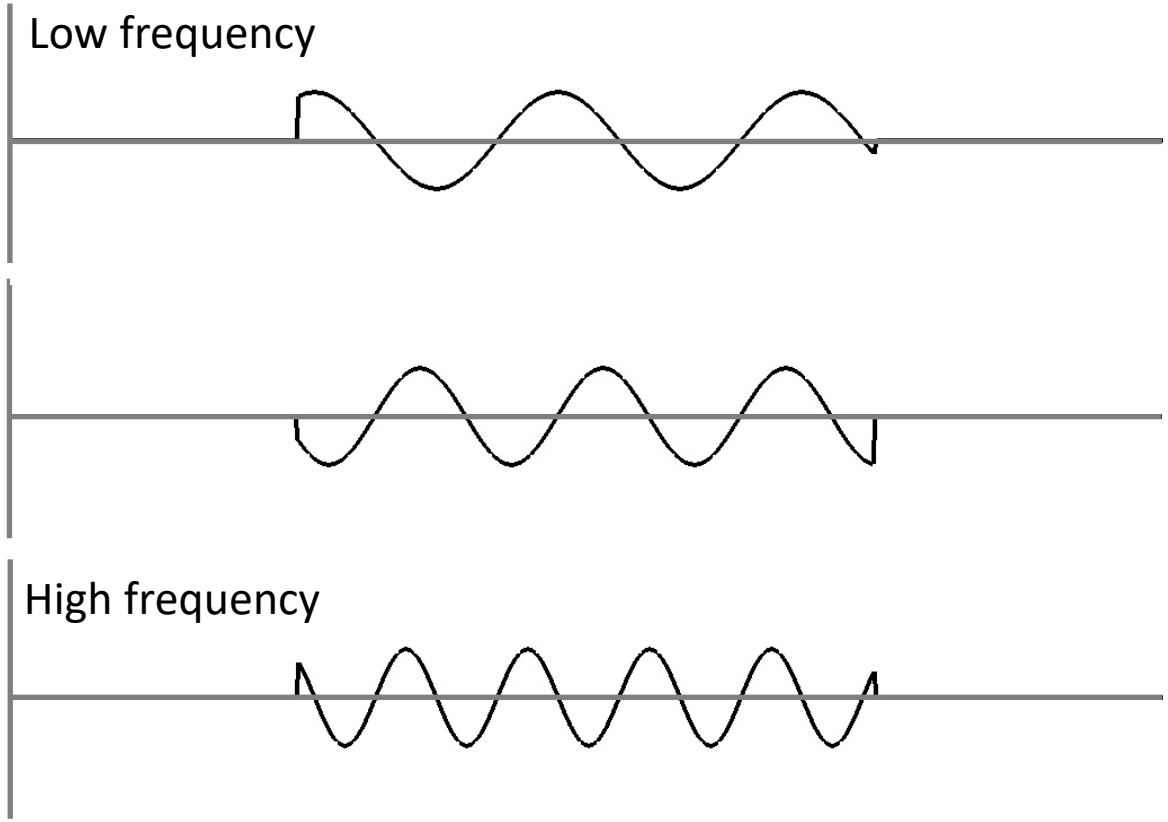
- Temporal span
- **Amplitude/gain**
  How loud is the sound?



Midterm project

The individual waves are represented in terms of their:

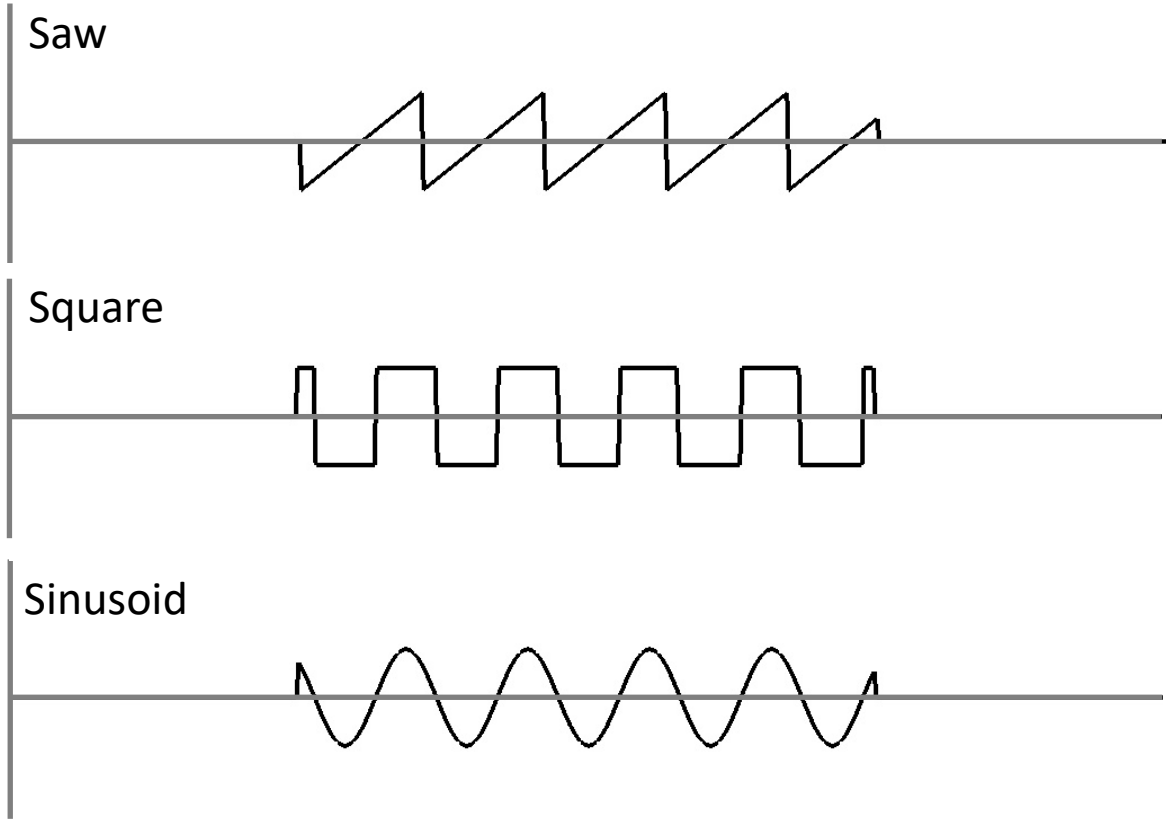
- Temporal span
- Amplitude/gain
- **Frequency**
  What is the pitch of the sound?



Midterm project

The individual waves are represented in terms of their:

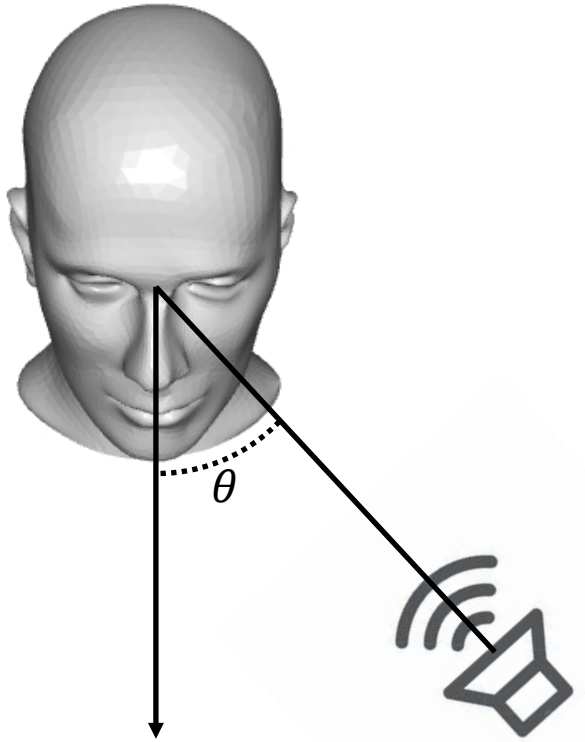
- Temporal span
- Amplitude/gain
- Frequency
- **Shape**
  What is the shape of the repeating part of the wave?

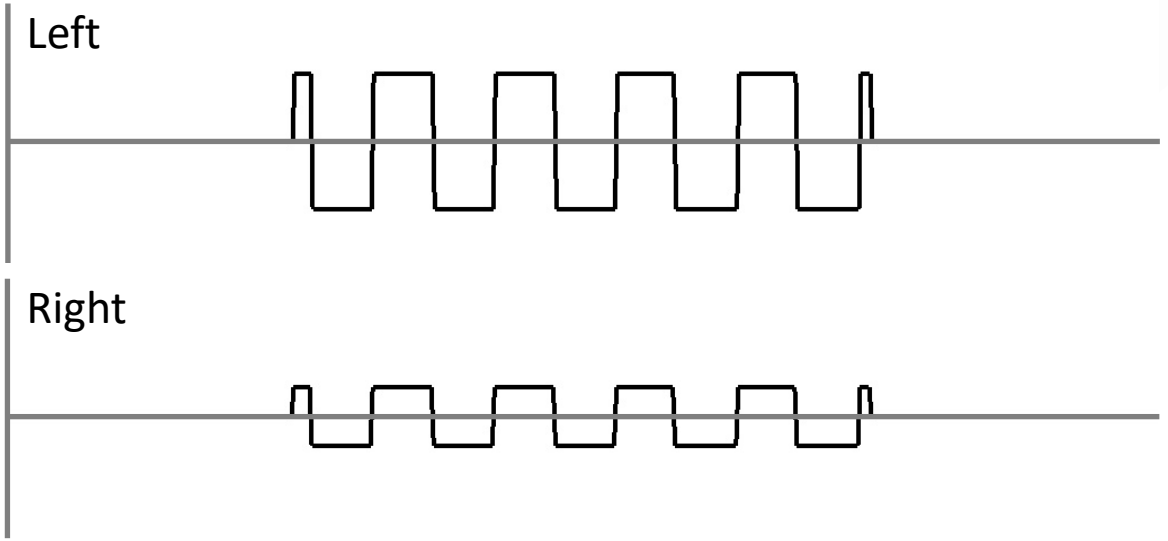


Midterm project

The individual waves are represented in terms of their:

- Temporal span
- Amplitude/gain
- Frequency
- Shape
- **Angle**
  How frontally aligned is the sound?





Midterm project

Write out audio signals to `.wav` files, which includes:

- Header:
  Describes the audio signal (number of samples, sampling rate=44.1KHz, number of channels=2, etc.)

```
duration = num_samples /44100;
```

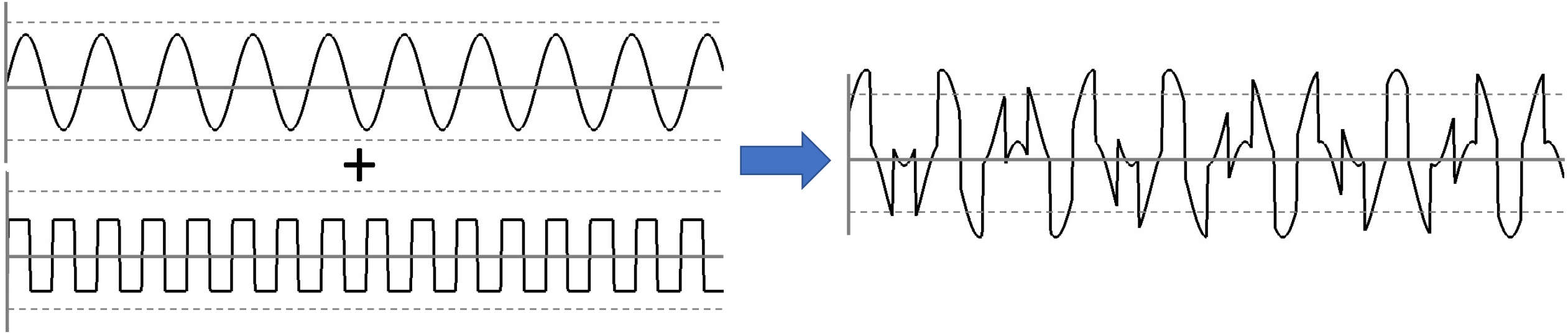
- Audio content:
  The interleaved samples of the left and right signals (in binary)
  - Total number of samples is `2*num_samples`, with the even-indices sampling the signal for the left ear and the odd-indices sampling the right.
  - Each sample is represented with a 16-bit integer type (`int16_t`).

Midterm project

[WARNING]

When writing the signal to a `.wav` file, each sample of the signal is represented with a 16-bit int type (`int16_t`).

⇒ Even if the signal samples are within bounds, their sum may not be.



Midterm project

[WARNING]

When writing the signal to a `.wav` file, each sample of the signal is represented with a 16-bit int type (`int16_t`).

⇒ Even if the signal samples are within bounds, their sum may not be.

⇒ Be sure to clamp to avoid overflow happens!!!