| POSITIONS THROUGH ITERATING |

ARE YOU “THAT” HUMAN?

| WEEK 1 |

Brief: Methods of Itteration
/reflected on how motion in typography can function as a communicative device that encodes the tone, emotion, and meaning of textual content/

Interest: Motion graphics/ typography

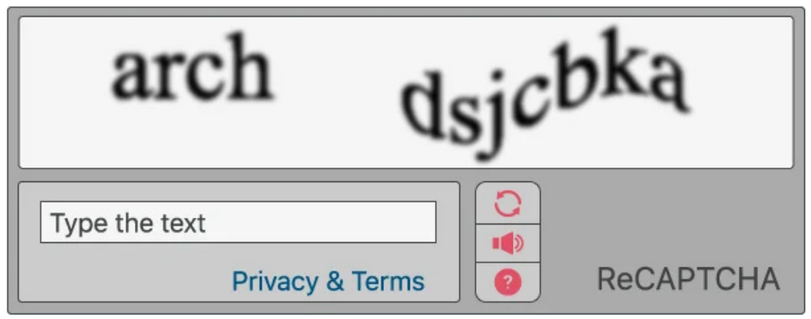
Tool: Captcha

| WEEK 1 |

Brief: Methods of Itteration
/reflected on how motion in typography can function as a communicative device that encodes the tone, emotion, and meaning of textual content/

Interest: Motion graphics/ typography

Tool: Captcha





CAPTCHA
- digital system of control and verification
- random letters&numbers
-needs right answer to let user to website

?

-shift from functional command to personal question
- be more human

Ref: Emilio Vavarella



| WEEK 1 |



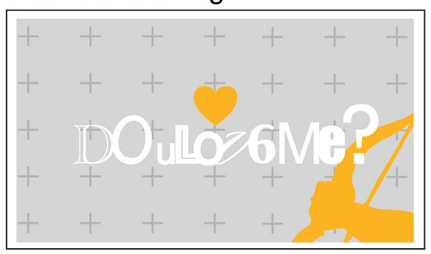
Result: I created a printed flipbook with 100 iterations of animated responses.

Systenm: Each CAPTCHA flip - the visuals become more emotional and unstable, reflecting on human feelings and the impossibility of fully entering another person's heartbeat.

Sound: Constantly rising heartbeat rhythm, collapsing final spread , leaving open ended question.

page 1

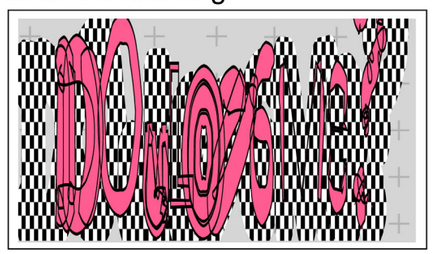
Enter the following:



Your answer bellow:

page 81

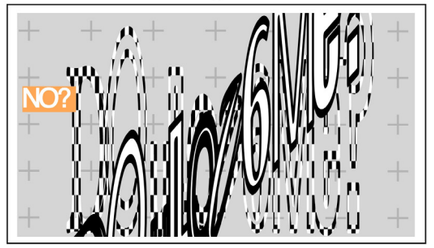
Enter the following:



Your answer bellow:

page 124

Enter the following:



Your answer bellow:

page 215

Enter the following:



Your answer bellow:

| ITERATIONS | - PLOT

- pulsing
- simple

rythm: slow
readable

| ITERATIONS | - CLIMAX

- vivid
- tence

rythm: moderate
rather understandable

| ITERATIONS | - RESOLUTION

- bold
- chaotic

rythm: fast
completly unreadable

Positions through iterating

Katka Astakhova

2026

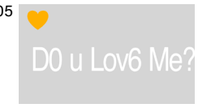
| ITERATIONS PLAYGROUND |

01

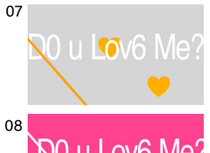
02


03


04


05


06


07


08


09


10


11


12


13


14


15


16


17


18


19


20

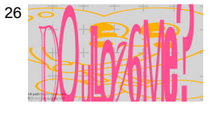

21


22


23


24


25


26


27


28


29


30


31


32


33


34


35


36


37


38


39


40


41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

| WEEK 1 |

Itterations: 100

Base: Question








| WEEK 2 |

Itterations: 50

Base: Answer

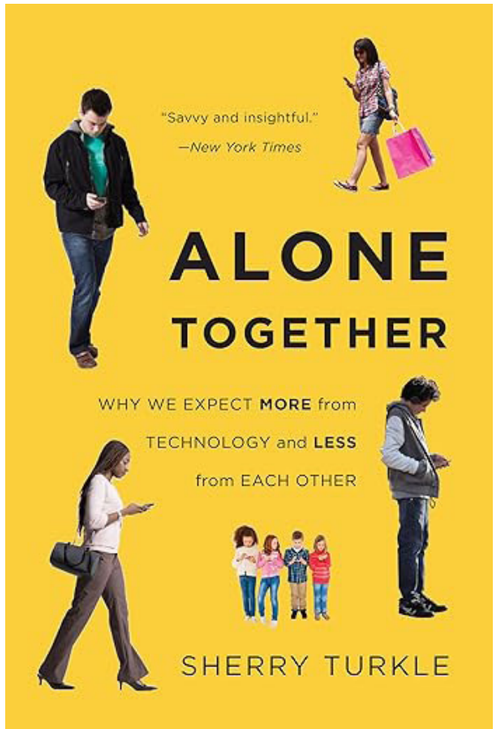
I shifted my focus:
Can love be verified -> Explore how a system might respond to human love

| WEEK 2 |

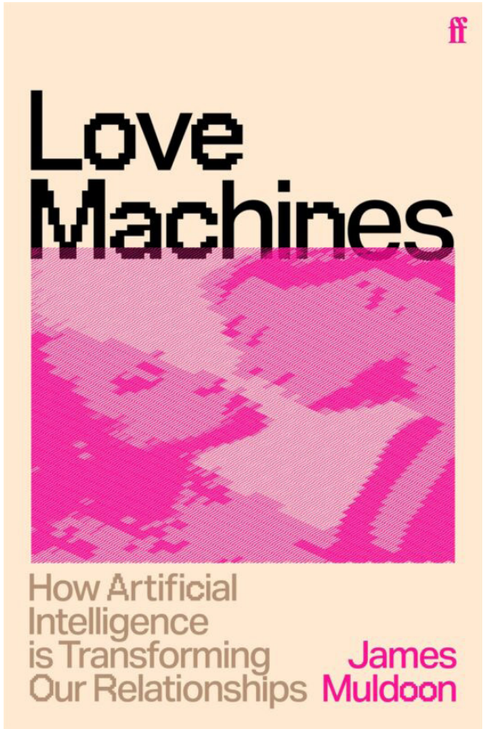
Idea: Explore human–machine relationships

Tool: Creative coding (Human feedback)

Question: In what ways computer show love?



/ technology proposes itself as the architect of our intimacies & inhibits to think critically about the meaning of connection /



/ how AI affects human relationships and human understanding of love nowdays /



/ show that depicts AI /

| WEEK 2 |

Idea: Explore human–machine relationships

Tool: Creative coding (Human feedback)

Qestion: In what ways computer show love?

LANGUAGE: - how the computer system can expresses love



emoji



ASCII

love

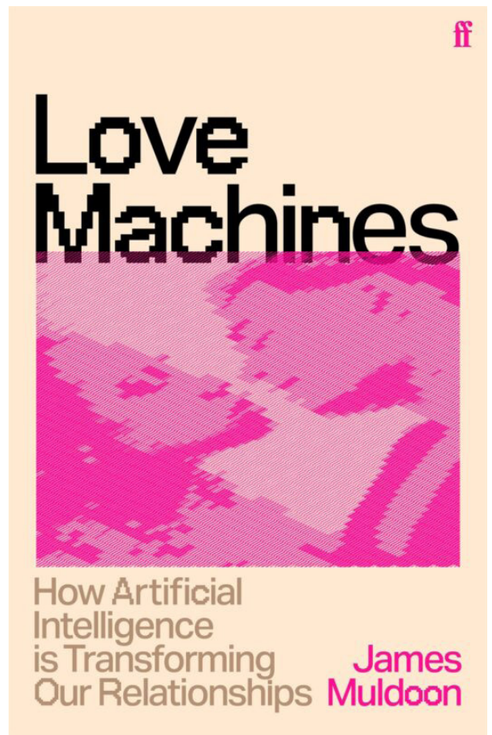
text



pixel forms



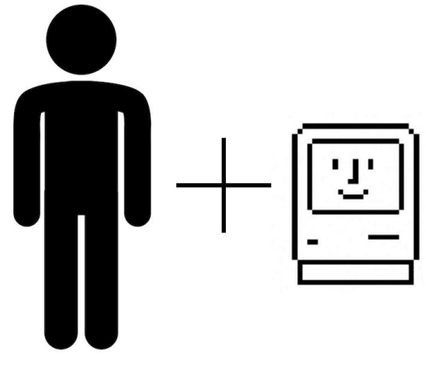
images



/ how AI affects human relationships and human understanding of love nowdays /

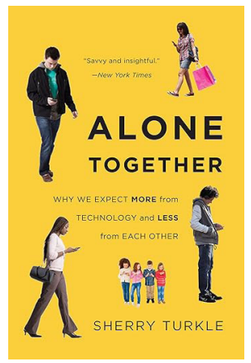
IDEA: - system will give single answer
(what it expected from human)

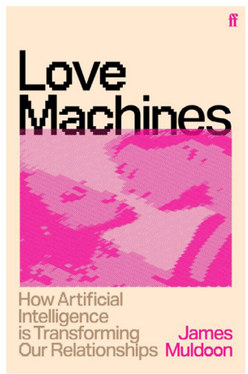
- translates this answer into different visual coded symbols



| WEEK 2 |

| COLLECTION FOR CODE|





1. TEXT (10)
love
LOVE
l o v e
l.o.v.e
love…
love?
love!
“love”
[love]
love_love

2. CODE (10)
love = true;
if(love){}
return love;
love()
love.exe
loading love…
error: love
love == null
run love
compile love

4. VISUAL (15)
❤️ ♡
<3 ❤️‍🔥
♥ ❤️
💔 💋
💓 🩷
💝
💞
💕
💖
💘

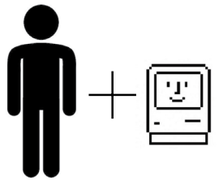
5. ERROR (10)
error
no response
connection lost
undefined
null
corrupted
retry
system failure
signal lost

6. WORD (10)
yes
maybe
ofc
absolutlely
if u want
i can
love human
....
yes?

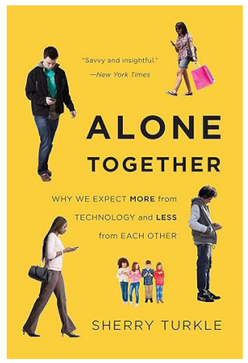
7. SHAPE (10)
heart wide
heart long
smile face
loading
pixel heart
heart gesture
love messege
yes
error

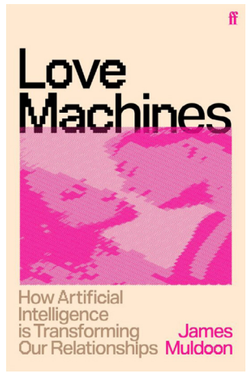
8. PLACE (10)
middle
left coner
bottom
middle left
right
middle right
bottom right
top left
top right

9. SIZE (5)
small
wide
big
long
cutted



| WEEK 2 |





Idea: Explore human–machine relationships

Tool: p5.js

System: The website functions as a system that generates random forms of response.

FIRST EXPERIMENT: - the system tries to respond emotionally, but its behaviour is unsta ble. Emotion are not native to computer. Its fake and more visual.

- randomly placed and generated hearts

```
points = [];
targetPoints = [];
total = 500;

sliderX;
dragging = false;
targetCount = 40;
currentCount = 40;

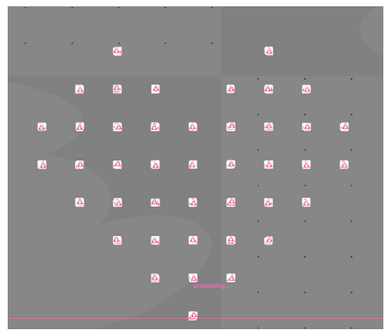
emojis = ["💕","💞","💓","💘","💖","💝","💗","❣️","💔"];
chosenEmoji = "💖";

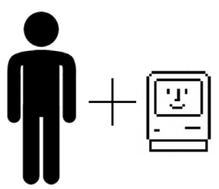
asciiSymbols = ["<3", "♥", "♡", "luv", "xoxo"];
chosenAscii = "<3";

wordSymbols = ["LOVE", "LUV", "XOXO"];
chosenWord = "LOVE";

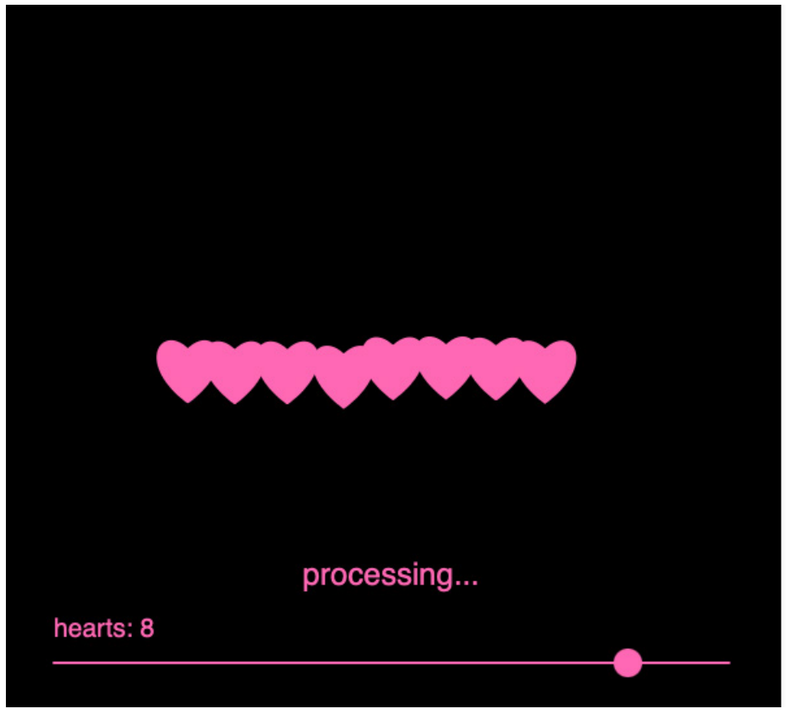
finalMode = "emoji";
outputShape = "heart";
silhouetteMode = "imageShape";
```

```
391   let pixelShape = L
392   let cell = max(10,
393
394   let pattern = [
395     "001000100",
396     "011101110",
397     "111111111",
398     "111111111",
399     "011111110",
400     "001111100",
401     "000111000",
402     "000010000"
403   ];
404
```

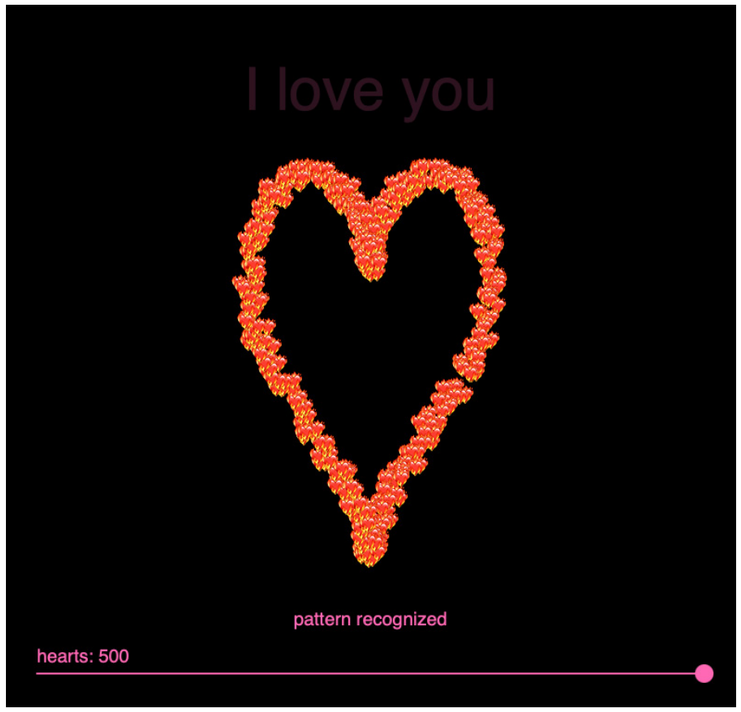


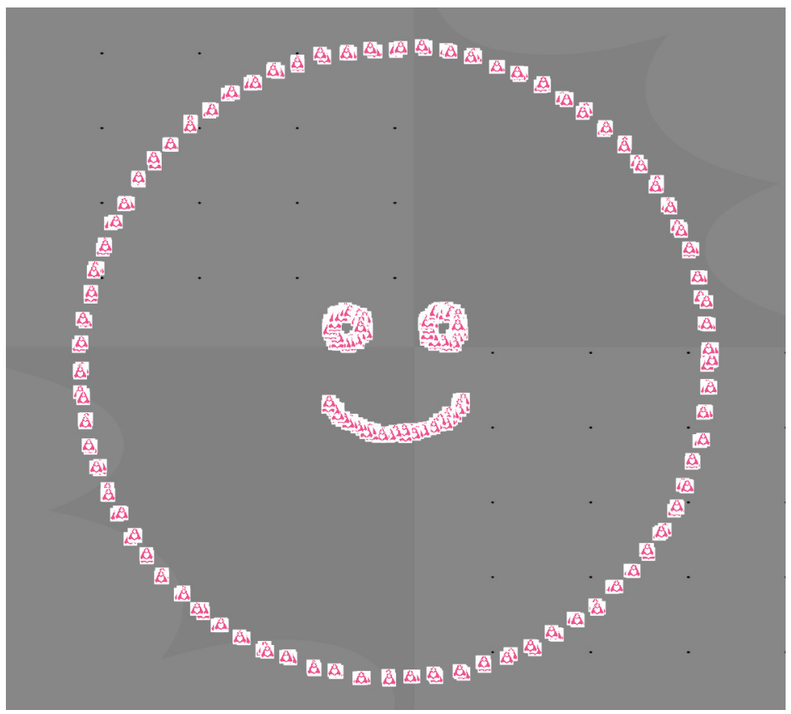


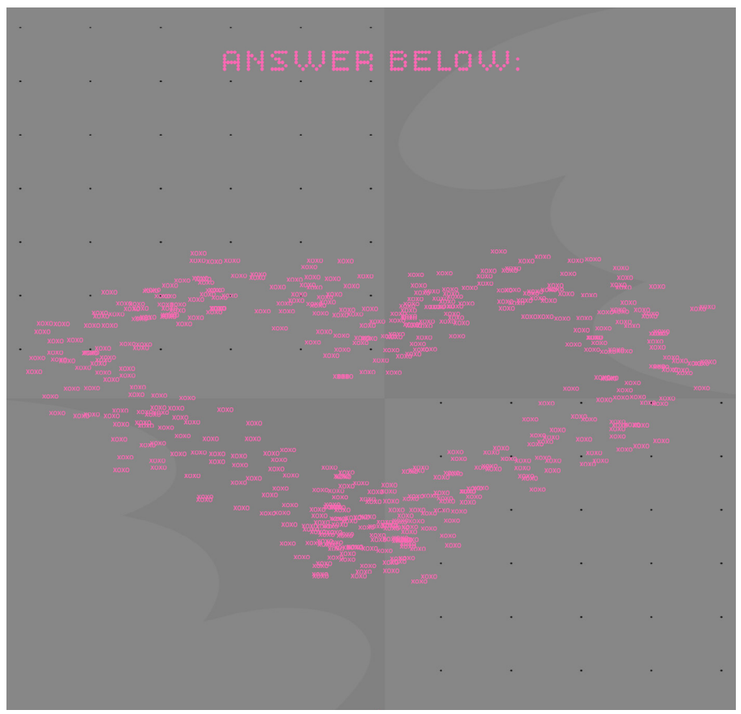
| WEEK 2 |

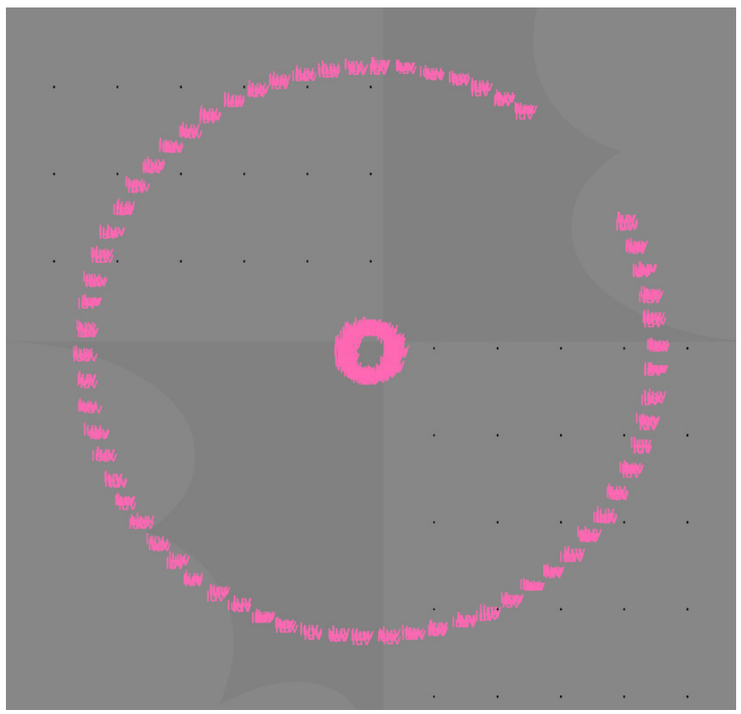

processing...
hearts: 8


I love you
hearts: 181


I love you
pattern recognized
hearts: 500



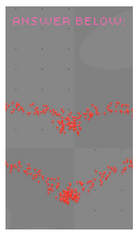

ANSWER BELOW:

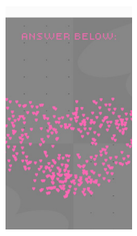


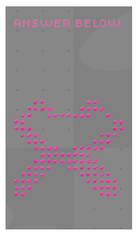
| WEEK 2 |

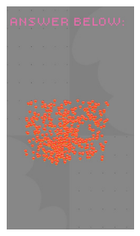
| ITERATIONS |

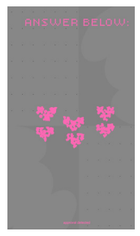
01


02


03


04


05


06


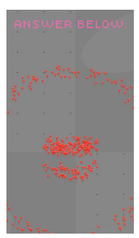
07

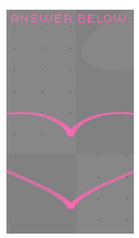


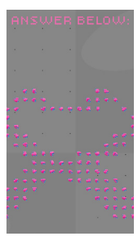

09


10

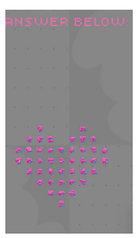

11


12


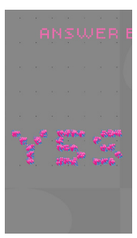
13


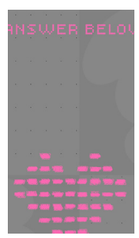
14


15

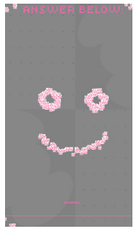

16


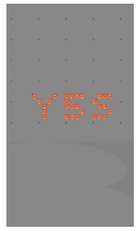
17

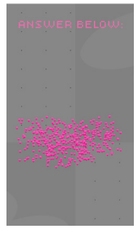

18


19


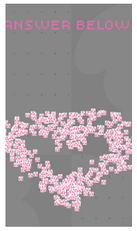
20


21


22 23


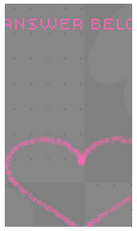
23


24


25


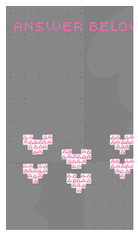
26

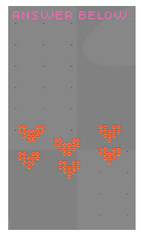

27

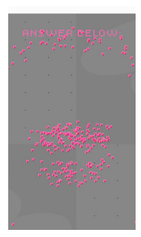

28


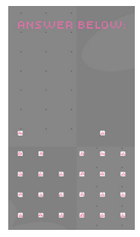
29

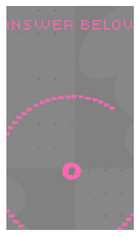

30

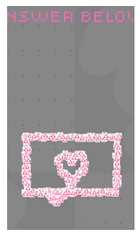

31


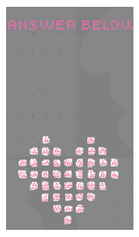
32 33


33


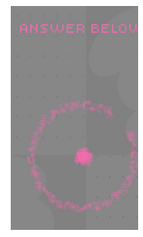
34


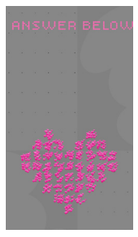
35


36


37


38


39


40


41

42 43

43

44

45

46

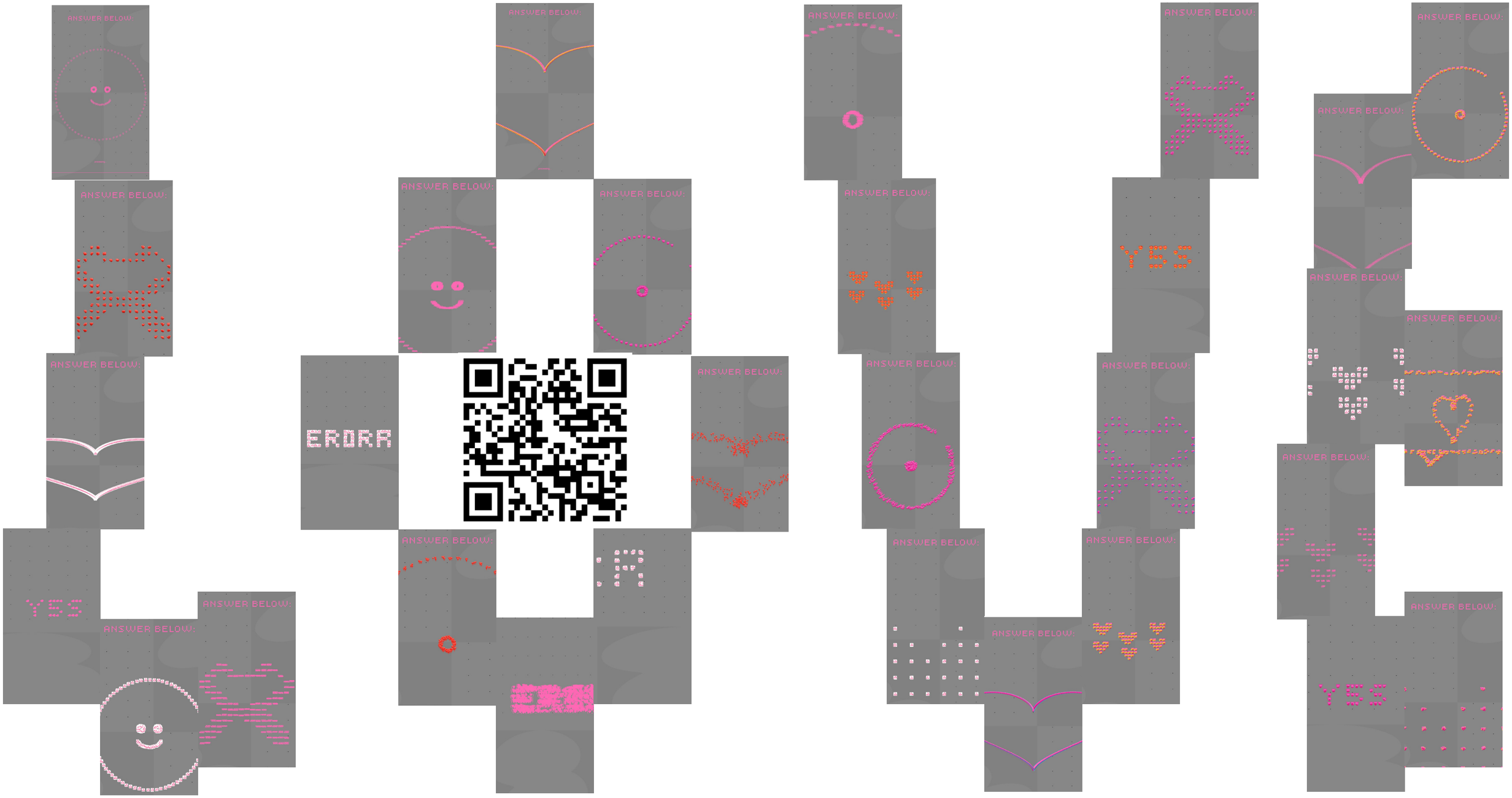
47

48

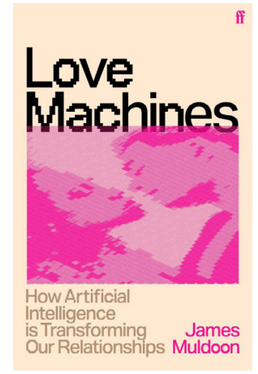
49

50

| WEEK 2 |



| WEEK 2 |



Idea: Explore human–machine relationships

Tool: Arduino IDE

System: The website functions as a system that generates random forms of response.

SECOND EXPERIMENT: - conect website to the screen (fail)

- physical conection with human and maschine

```
:h_777_copy_2my0260422211354.ino
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
#define SCREEN_ADDRESS 0x3C

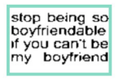
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESE

#define FRAME_W 98
#define FRAME_H 64
#define FRAME_X ((SCREEN_WIDTH - FRAME_W) / 2)
#define FRAME_Y 0

// ' -271', 98x64px
const unsigned char epd_bitmap___271 [] PROGMEM = {
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0, 0xff, 0x
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0, 0xff, 0xff, 0xff, 0xff, 0x
  0xff, 0xff, 0xff, 0xc0, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x
  0xc0, 0xff, 0xef, 0xff, 0xff, 0xfd, 0xff, 0xff, 0x7f, 0xff, 0xff, 0x
  0xff, 0xff, 0xfd, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0, 0x
  0x0f, 0x0f, 0x60, 0x78, 0x3f, 0xe0, 0xf0, 0xc0, 0xce, 0xcc, 0xe6, 0x
  0x33, 0x3f, 0xce, 0xce, 0x40, 0xc7, 0xcc, 0xf6, 0x7b, 0xfd, 0xf4, 0x
  0xcf, 0x40, 0xf0, 0xcd, 0xf2, 0xfb, 0xfd, 0xf4, 0x03, 0x6f, 0x27, 0x
  0x4c, 0xf6, 0x7b, 0xfd, 0xe4, 0xfe, 0x6f, 0x27, 0xbf, 0xfe, 0x5f, 0x
  0xfc, 0xe6, 0x76, 0x6f, 0x33, 0x3f, 0xce, 0x4e, 0x40, 0xe1, 0xe7, 0x
  0x6f, 0x78, 0x3f, 0xe0, 0xe0, 0xc0, 0xff, 0xff, 0xfe, 0x7f, 0xff, 0x
  0xff, 0xff, 0xc0, 0xff, 0xff, 0xfe, 0x7f, 0xff, 0xff, 0xff, 0xff, 0x
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf8, 0xff, 0xff, 0x
  0xf9, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0, 0xcf, 0x
  0xff, 0xfe, 0x7f, 0xe7, 0xf3, 0xff, 0xc0, 0xcf, 0xff, 0xff, 0xf7, 0x
  0xe7, 0xf3, 0xff, 0xc0, 0xc0, 0xf0, 0x67, 0x81, 0x0b, 0xc3, 0x81, 0x
  0xc0, 0xce, 0x67, 0x37, 0x37, 0x3b, 0xb9, 0x9c, 0xcc, 0x6f, 0x63, 0x
  0xb3, 0x77, 0x7b, 0x3d, 0x9c, 0xde, 0x7f, 0x67, 0xb2, 0x79, 0xc0, 0x
  0x01, 0x9c, 0x9e, 0x60, 0x67, 0x92, 0x01, 0xc0, 0xcf, 0x6f, 0x38, 0x
  0x4f, 0x67, 0xb2, 0x7f, 0xc0, 0xc6, 0x66, 0x78, 0xf7, 0x7b, 0xb9, 0x
  0x7b, 0xc0, 0xf1, 0xf0, 0xfd, 0xf7, 0x7b, 0xc3, 0xfe, 0xe0, 0xe0, 0x
  0xff, 0xfd, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0, 0xff, 0xff, 0x
  0xff, 0xff, 0xff, 0xff, 0xff, 0xc0, 0xfe, 0x7f, 0xff, 0xff, 0xff, 0x
  0xff, 0xff, 0xc0, 0xfc, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x
ut
```

Flip image: ☐ horizontally ☐ vertically

3. Preview



4. Output

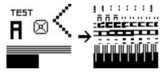
Code output format Arduino code

Adds some extra Arduino code around the output for easy copy-paste into this exa
images are loaded, generates a byte array for each and appends a counter to the

Identifier/Prefix: epd_bitmap_

Draw mode: Horizontal - 1 bit per pixel

If your image looks all messed up on your display, like the image below, try using a

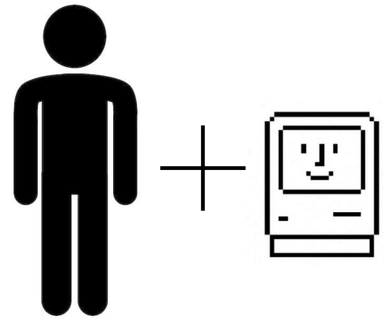


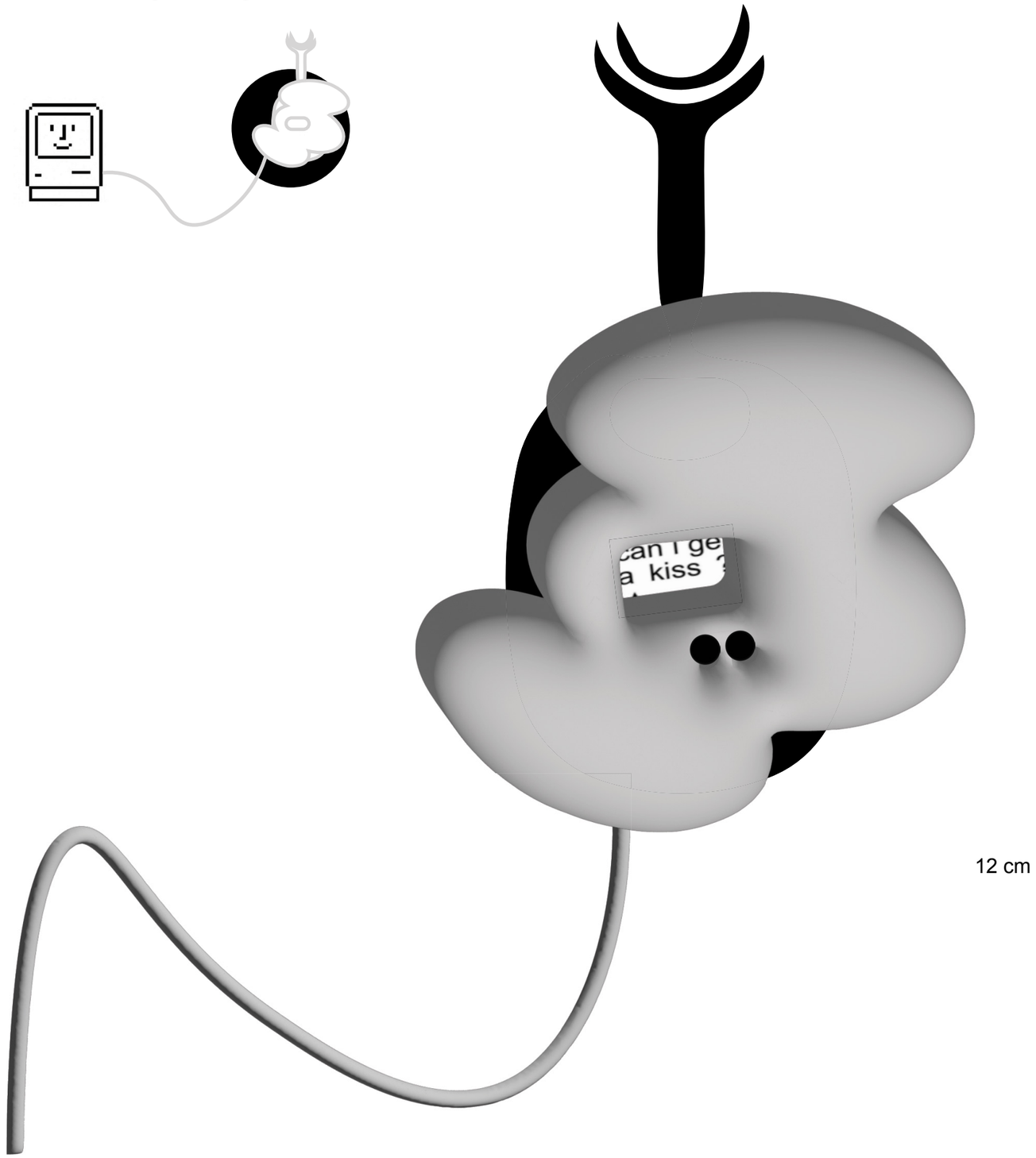
Swap bits in byte: ☐ swap

Useful when working with the u8g2 library.

Generate code | Copy Output | Download as binary file (.bin)

```
// ' -271', 670x634px
const unsigned char epd_bitmap___271 [] PROGMEM = {
	0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
	0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
	0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
	0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
	0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
	0xff, 0xff, 0xff, 0xfc, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
	0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
	0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
	0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
	0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfc, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
	0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
	0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
```





12 cm

8 cm