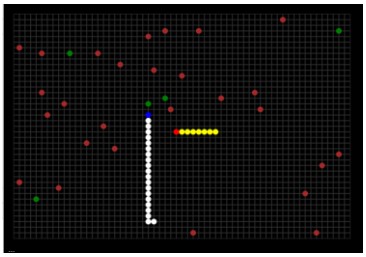
GLIZDA – gra (26)



Glizdą poruszamy po planszy za pomocą klawiszy ze strzałkami.
Glizda zjada jabłka: jeżeli zielone, to rośnie jej ogon, jeżeli brązowe to ogon się kurczy.
Glizda może strzelać fragmentem ogona i wtedy strzał zjada wszystko do końca planszy.
Po planszy porusza się również automatyczna glizda, która zjada kolejne jabłka.
Jeżeli jabłka się skończą, losowana jest kolejna partia.

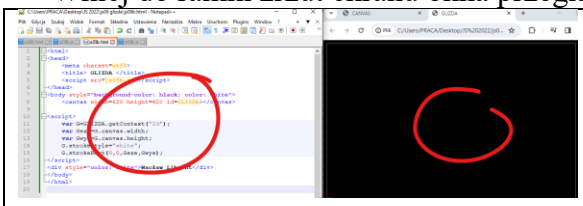
Plansza (2)

- W swoim folderze utwórz 2 nowe dokumenty: **js09.html** i **js09.js**
- Otwórz oba dokumenty w notatniku, a dokument HTML w przeglądarce
- Do dokumentu **HTML** wklej tekst z ramki

```
<html>
<head>
	<meta charset=utf8>
	<title> GLIZDA </title>
	<script src=js09.js></script>
</head>
<body style="background-color: black; color: white">
	<canvas width=620 height=420 id=GLIZDA></canvas>

<script>
	var G=GLIZDA.getContext("2d");
	var Gsze=G.canvas.width;
	var Gwys=G.canvas.height;
	G.strokeStyle="white";
	G.strokeRect(0,0,Gsze,Gwys);
</script>
</body>
</html>
```

tło strony czarne i kolor znaków biały
obszar canvas ma rozmiar 620x420 pikseli
dostęp do obszaru za pomocą „uchwytu" o nazwie G
zadeklarowane dwie zmienne Gsze i Gwys, w których zapamiętujemy szerokość i wysokość obszaru canvas
rysujemy białą ramkę na granicach canvas

- Zmień tytuł strony **GLIZDA** na swoje **inicjały**
- Zapisz dokumenty i odśwież przeglądarkę
  powinna pojawić się ramka o rozmiarze 620x420 pikseli
- Wklej do ramki zrzut ekranu okna przeglądarki i dokumentu HTML



Pole gry (2) – szachownica z kwadratów

*Plansza, po której porusza się glizda składa się KOLxWIE kwadratów o boku 2*PRO*
Ponieważ glizda będzie poruszać się po skrzyżowaniach, więc możliwych pozycji jest KOL+1 (0..60) i WIE+1 (0..40)
Kwadraty rysujemy ciemnoszarym kolorem

- Do dokumentu JS wklej tekst z ramki

```
function FPlansza(){
	var bok=2*PRO;
	G.clearRect(0,0,Gsze,Gwys);
	G.strokeStyle="rgb(50,50,50)";
	for (var k=0;k<KOL;k++){
		for (var w=0;w<WIE;w++){
			G.strokeRect(bok+k*bok,bok+w*bok,bok,bok);
		}
	}
```

```
        G.fillStyle="white";
        G.fillText("Libront Wacław",0,8);
}
```

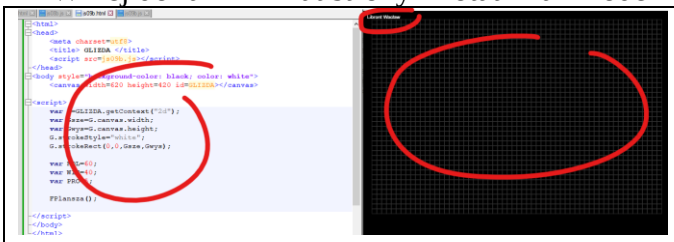
czyścimy obszar canvas
ustalamy ciemnoszary kolor
w pętlach FOR rysujemy kwadraty odległe od brzegów o bok, żeby glizda się dobrze rysowała
w lewym górnym rogu wypisujemy nazwisko i imię

- Do dokumentu HTML, przed znacznik </scrpt> wpisz instrukcje z ramki

```
var KOL=60;
var WIE=40;
var PRO=5;

FPlansza();
```

*zmienne główne programu: kolumny, wiersze, promień koła (bok=2*promień)*

- Zmień **nazwisko i imię** na swoje
- Wklej do ramki zrzut strony i notatnika z kodem HTML



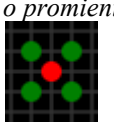
Kolorowe koło (jabłko, głowa, ogon) (2)

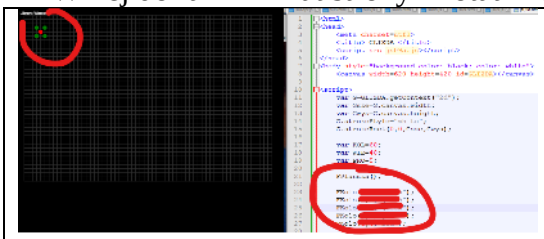
Kolorowe koła, to główne elementy gry

- Do dokumentu JS wklej tekst z ramki

```
function FKolo(kol,wie,kolor){
        var bok=2*PRO;
        G.beginPath();
        G.fillStyle=kolor;
        G.arc(bok+kol*bok,bok+wie*bok,PRO,0,2*Math.PI);
        G.fill();
}
```

w rogach kwadratów (na skrzyżowaniach) rysowane jest kolorowe koło o promieniu PRO

- W dowolnym miejscu planszy narysuj 5 kół, jak na rysunku 
  pierwsze koło narysuj poleceniem `FKolo(10,10,"green");`
- Wklej do ramki zrzut strony i notatnika z kodem HTML



Latająca głowa glizdy (2)

Glizda ma czerwoną głowę, która porusza się w prawo
Gdy glizda osiągnie brzeg planszy, pojawia się po drugiej stronie

- Do dokumentu HTML, przed funkcję **Fplansza()** wpisz zmienne z ramki

```
var SKOK=100;
var CZAS;

var Gkol=10;
var Gwie=10;
var GkolV=1;
var GwieV=0;
```

główne zmienne animacji CZAS i SKOK
współrzędne głowy glizdy (Gkol, Gwie) i szybkość (GkolV,GwieV)
glizda porusz się w prawo, bo szybkość kolumnowa=1, a szybkość wierszowa=0

- Do dokumentu JS wklej tekst z ramki

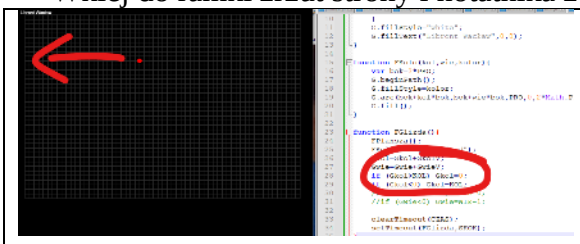
```
function FGlizda(){
	FPlansza();
	FKolo(Gkol,Gwie,"red");
	Gkol=Gkol+GkolV;
	Gwie=Gwie+GwieV;
	if (Gkol>KOL) Gkol=0;

	clearTimeout(CZAS);
	setTimeout(FGlizda,SKOK);
}
```

główna pętla animacyjna
rysujemy planszę złożoną z szarych kwadratów
rysujemy czerwoną głowę na skrzyżowaniu (Gkol,Gwie)
obliczamy nowe współrzędne głowy na podstawie (GkolV,GwieV)
ponieważ glizda porusza się w prawo (GkolV=1), to sprawdzamy przekroczenie prawego brzegu planszy

- Do dokumentu HTML, przed znacznik </script>, wpisz wywołanie funkcji `FGlizda();`

uruchomienie pętli animacyjnej

- Glizna porusza się w prawo - **zmień kierunek** poruszania się w lewo za pomocą zmiennej **GkolV**
- Wpisz do funkcji **Fglizda()** instrukcję sprawdzającą lewy brzeg `if (Gkol<0) Gkol=KOL;`
- Wklej do ramki zrzut strony i notatnika z kodem JS



Klawiszologia (2)

onkeydown, ponieważ funkcja, którą można podpiąć pod to zdarzenie często będzie przyjmowała dodatkowy argument. Argument ten będzie obiektem zawierającym przydatne informacje na temat tego, jaki przycisk został wciśnięty przez użytkownika. Ważne jest, że aby to zdarzenie zostało wywołane konieczne jest uzyskanie przez dany element strony fokusa, czyli obsługi klawiatury

- Do dokumentu HTML, za znacznikiem **</canvas>**, wpisz znaczniki

```
<br>
<label id=idKL></label>
```

etykieta, w której będzie wypisywany wciśnięty kawisz

- Do dokumentu HTML, przed funkcją **FGlizda()**, wklej tekst z ramki

```
document.addEventListener("keydown", WcisnietyKlawisz,false);
document.addEventListener("keyup", PuszczonyKlawisz,false);
```

obsługa zdarzeń związanych z klawiaturą
w zmienej KLA
jeżeli wciśnięty klawisz, to wywołana zostanie funkcja WcisnietyKlawisz
jeżeli puszczono klawisz, to wywołana zostanie funkcja PuszczonyyKlawisz

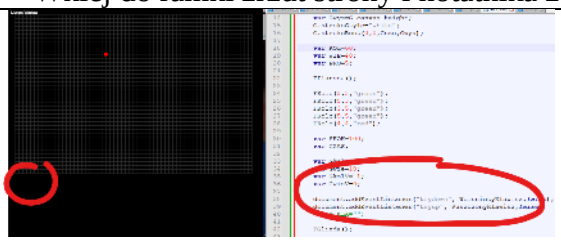
- Do dokumentu JS wklej tekst z ramki

```
function WcisnietyKlawisz(e){
	var KLA=e.key;
	idKL.innerHTML=KLA;
```

```
}

function PuszczonyKlawisz(e){
      var KLA="...";
      idKL.innerHTML=KLA;
}
```

w zmiennej KLA zapamiętujemy ops klawisza
opis klawisza wstawiony to etykiety idKL

- „Pobaw się” chwilę klawiaturą i sprawdź opisy klawiszy
- Wklej do ramki zrzut strony i notatnika z kodem JS



Sterowanie glizdą (2)

Po wciśnięciu klawiszy ze strzałkami, głowa glizdy zaczyna poruszać się w wybranym kierunku, aż do momentu kolejnej zmiany kierunku.

- Do dokumentu JS, do funkcji **WcisnietyKlawisz**, wpisz instrukcje z ramki

```
if (KLA=="ArrowLeft") {GkolV=-1; GwieV=0;}
if (KLA=="ArrowRight") {GkolV=1; GwieV=0;}
if (KLA=="ArrowUp") {GwieV=-1; GkolV=0;}
if (KLA=="ArrowDown") {GwieV=1; GkolV=0;}
```

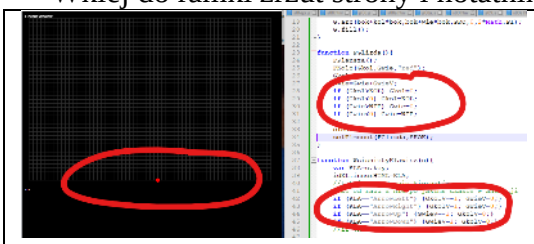
jeżeli wciśnięto klawisz ze strzałką, to ustawiamy szybkość tylko w tym kierunku, drugi zerujemy

- Do dokumentu JS, do funkcji **FGlizda**, wpisz instrukcje z ramki

```
if (Gwie>WIE) Gwie=0;
if (Gwie<0) Gwie=WIE;
```

sprawdzanie prawego i lewego brzegu już powinieneś mieć
dopisz sprawdzanie przejścia przez dolny i górny brzeg

- Ustaw glizdę tak, aby poruszała się wzdłuż dolnej krawędzi planszy
- Wklej do ramki zrzut strony i notatnika z kodem JS



Losowanie jabłek (2)

Poruszająca się glizda będzie zżerać dojrzałe zielone jabłka, co spowoduje przyrost ogona (albo brązowe – zmniejszanie)
Położenie jabłek zostanie rozlosowane – za każdym razem 10 zielonych i 10 brązowych
Pozycje jabłek powinny znaleźć się w tablicy dwuwymiarowej, która odzwierciedlać kratki planszy
W każdej komórce tablicy zapisujemy kolor jabłka; „” – bez jabłka

- Do dokumentu HTML, przed funkcję **FGlizda**, wpisz instrukcje z ramki

```
var JAB=[];
var IleJab=10;
FZerujJablka();
FLosujJablka(IleJab,"green");
```

JAB *tablica dwuwymiarowa (po ustawieniu) zawierająca rozlosowane jabłka*
IleJab *liczba jabłek w pojedynczym losowaniu*
FZerujJablka *funkcja ustawiająca tablicę dwuwymiarową i zerująca wszystkie komórki*
FlosujJablka *losowanie 10 zielonych jabłek*

- Do dokumentu JS, do funkcji **FGlizda**, za poleceniem **Fplansza();**, wpisz wywołanie funkcji z ramki

```
FRysujJablka();
```

FrysujJablka() *wrysowanie na planszę jabłek, w pętli animacyjnej będzie się automatycznie aktualizować stan jabłek*

- Do dokumentu JS, wklej tekst z ramki

```
function FZerujJablka(){
	JAB=[];
	for (var k=0;k<=KOL;k++){
		JAB[k]=[];
		for (var w=0;w<=WIE;w++){
			JAB[k][w]="";
		}
	}
}

function losowa(p,k) {
  return Math.floor(Math.random()*(k-p+1)+p);
}

function FLosujJablka(ile,kolor){
	var i=ile;
	do {
		k=losowa(0,KOL);
		w=losowa(0,WIE);
		if (JAB[k][w]==””) {
			i=i-1;
			JAB[k][w]=kolor;
		}
	} while (i>0)
}

function FRysujJablka(){
	for (var k=0;k<=KOL;k++){
		for (var w=0;w<=WIE;w++){
			var kol=JAB[k][w];
			if (kol!="") FKolo(k,w,kol);
		}
	}
}
```

FZerujJablka

w pierwszej pętli tworzymy kolejne kolumny tablicy JAB – 41 komórek
w drugiej pętli, do w każdej komórce kolumny tworzymy 61 komórek wierszy
do każdej komórki wpisujemy napis pusty „ ”

FLosowa *całkowita liczba losowa z przedziału <p,k>*

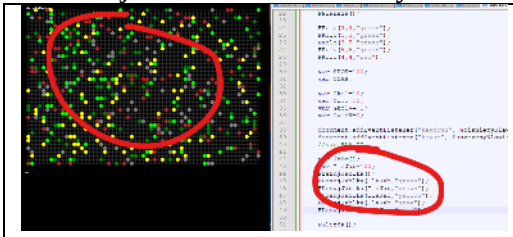
FLosujJablka(ile,kolor)

nie możemy zwykłej pętli for, bo gdy wylosujemy komórkę, w której już jest jabłko? – dlatego pętla logiczna do…while
losujemy dwie współrzędne i sprawdzamy, czy jest pusta
jeżeli tak, to zmniejszamy licznik jabłek do rozlosowania i w komórce wpisujemy kolor
powtarzamy losowanie współrzędnych aż rozlosowane wszystkie jabłka

FRysujJablka

w dwóch pętla FOR przelatujemy przez wszystkie komórki tablicy JAB
jeżeli jest ustawiony kolor (różne od pustego), to rysujemy jabłko w tym kolorze

- Polecenie `FLosujJablka(IleJab,"green");` losuje 10 zielonych jabłek
  Za pomocą **tej funkcji rozlosuj jeszcze** narysuj **6 razy po 100 jabłek** w różnych kolorach
- Wklej do ramki zrzut strony i notatnika z kodem HTML



Przycisk losowania (2)

Zautomatyzowane losowanie będzie nam potrzebne w dalszym etapie pracy nad programem

- Do dokumentu JS, wpisz nową funkcję z ramki

```
function FUstawJablka(){
    FLosujJablka(IleJab,"green");
    FLosujJablka(IleJab,"brown");
    FRysujJablka()
}
```

- Do dokumentu HTML, za definicją etykiety **idKL**, wpisz znaczniki z ramki

```
<br>
<input type=button value=LOSUJ onClick=FUstawJablka()>
```

przycisk, za pomocą którego rozlosujemy różne kolory jabłek

- W dokumencie HTML

  - ustaw zmienną `var IleJab=20;`
  - usuń losowanie 6 gatunków jabłek

  losowanie zielonych i brązowych jabłek
  przerysowanie tablicy z jabłkami

- Kliknij wiele razy przycisk LOSUJ, aby rozlosować 300-400 jabłek
- Wklej do ramki zrzut strony i notatnika z kodem JS

Konsumpcja jabłek (2)

Głowa glizdy wlezie na jabłko – zmieniamy długość ogona
zielone – ogon rośnie, brązowe – ogon maleje
jeżeli ogon o długości zero, to nie może już zmaleć

- Do dokumentu HTML, za definicją przycisku LOSUJ, wpisz znaczniki z ramki

```
<br>
Ogon: <label id=idOG></label>
```

etykieta, do której wpisujemy aktualną długość ogona

- Do dokumentu HTML, przed funkcją **FGlizda**, wpisz definicję zmiennej z ramki

```
var IleOgo=0;
```

zmienna, w której zapisujemy długość ogona, po zjedzeniu jabłka

- Do dokumentu JS, wklej tekst z ramki

```
function FSprawdzGlizde(){
      var kol=JAB[Gkol][Gwie];
      switch (kol) {
            case "green":
                  JAB[Gkol][Gwie]="";
                  IleOgo++;
            break;
            case "brown":
                  JAB[Gkol][Gwie]="";
                  IleOgo--;
            break;
      }
      idOG.innerHTML=IleOgo;
}
```

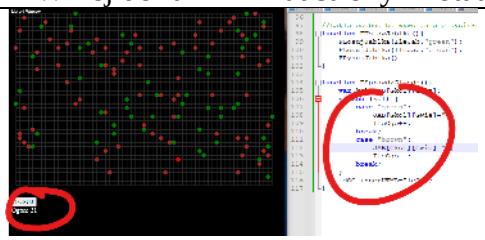
pobieramy kolor komórki tablicy, w której znajduje się głowa glizdy
za pomocą instrukcji warunkowej switch…case (rozbudowana wersja if …else decydujemy o długości ogona
zielony kolor: „zerujemy" komórkę tablicy i ogon zwiększa się o jeden
brązowy kolor: „zerujemy" komórkę tablicy i ogon zmniejsza się o jeden
na końcu wpisujemy długość ogona do etykiety

- W dokumencie JS, w funkcji **FGlizda**, za funkcją **FRysujJablka**, wpisz wywołanie funkcji

```
FSprawdzGlizde();
```

długość ogona sprawdzamy w pętli

- **Spróbuj zjeść 20 jabłek** – długość ogona 20
- Wklej do ramki zrzut strony i notatnika z kodem JS



Rysowanie ogona (2)

Ogon, to tablica OGO ze współrzędnymi miejsc, gdzie była głowa.
Po kazdym ruchu głowy, pola tablicy OGO przesuwają się o jedno miejsce

- Do dokumentu HTML, przed funkcją **FGlizda**, wpisz instrukcje z ramki

```
var OGO=[];
FUstawJablka();
```

OGO *tablica, w której zapisujemy współrzędne członów ogona*
FUstawJablka(); *po starcie już ustawione jabłka*

- Do dokumentu JS, wklej tekst z ramki

```
function FUstawOgon(){
      for (var i=IleOgo;i>=0;i--){
            OGO[i-1]=OGO[i-2];
      }
      OGO[0]={k:Gkol,w:Gwie};
}

function FRysujOgon(){
      for (var i=0;i<IleOgo;i++){
            FKolo(OGO[i].k,OGO[i].w,"yellow")
      }
}
```

FUstawOgon *za głową podąża ogon, kolejne komórki w OGO muszą zapamiętywać współrzędne głowy*
w pętli FOR przesuwamy komórki OGO od tyłu
ostatni komórka ginie
do pierwszej wstawiamy współrzędne głowy
FRysujOgon
w pętli FOR pobieramy współrzędne ogona
i rysujemy kółka w kolorze żółtym

- W dokumencie JS, w funkcji **FGlizda**, przed funkcją **FRysujJablka**, wpisz polecenie z ramki

```
FRysujOgon();
```

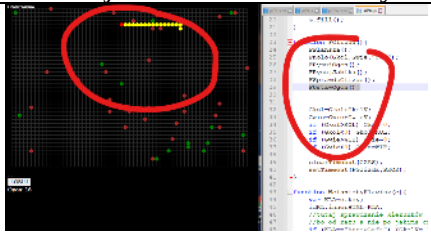
najpierw rysujemy głowę a potem ogon – w tej kolejności

- W dokumencie JS, w funkcji **FGlizda**, za funkcją **FSprawdzGlizde**, wpisz polecenie z ramki

```
FUstawOgon();
```

przestawiamy ogon dopiero po sprawdzeniu trafienia w jabłko – w tej kolejności

- Spróbuj zjeść 20 jabłek – długość ogona 20
- Wklej do ramki zrzut strony i notatnika z kodem JS



Glizda strzela ogonem (2)

Wciskamy spację i jeżeli glizda ma ogon, to oddaje strzał.
Fragment ogona leci 3 razy szybciej i kasuje wszystko po drodze.

- Do dokumentu HTML, przed funkcją **FGlizda**, wpisz instrukcje z ramki

```
var Skol=-2;
var Swie=-2;
var SkolV=0;
var SwieV=0;
var Sszy=3;
var Strz=false;
```

zmienne, za pomocą których kontrolujemy animację wystrzelonego ogona
(Skol, Swie) pozycja strzału
(SkolV,SwieV) kierunek strzału
Sszy szybkość poruszającego się ogona
Strz=false jeżeli fałsz, to ogon nie leci i można strzelać
jeżeli prawda, to ogon leci i nie można odać drugiego strzału, albo ogon zerowy i nie ma czym strzelać

- W dokumencie JS, w funkcji **WcisnietyKlawisz**, wpisz polecenie z ramki

```
if (KLA==" ") FGlizdaStrzal();
```

jeżeli spacja, to strzelamy ogonem

- W dokumencie JS, w funkcji **FGlizda**, za funkcją **FUstawOgon**, wpisz polecenie z ramki

```
FObliczStrzal();
```

sprawdzamy, co zestrzelił pocisk

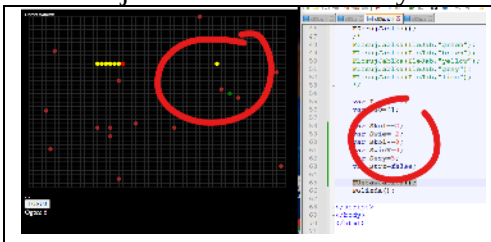
- Do dokumentu JS, wklej tekst z ramki

```
function FGlizdaStrzal(){
      if (Strz==false){
            if (IleOgo>0) {
                  Strz=true;
                  IleOgo--;
                  Skol=Gkol+GkolV;
                  Swie=Gwie+GwieV;
                  SkolV=GkolV;
                  SwieV=GwieV;
                  FKolo(Skol,Swie,"yellow");
            }
      }
}

function FObliczStrzal(){
      for (var i=0;i<Sszy;i++){
      if (Strz==true){
            var kol=JAB[Skol][Swie];
            if (kol!=""){
                  switch (kol) {
                  case "green":
                        JAB[Skol][Swie]="";
                        IleOgo++;
                        break;
                  case "brown":
                        JAB[Skol][Swie]="";
                        IleOgo--;
                        break;
                  }
                  FUstawOgon();
            }
            Skol=Skol+SkolV;
            Swie=Swie+SwieV;
            if (Skol>KOL) Strz=false;
            if (Skol<0) Strz=false;
            if (Swie>WIE) Strz=false;
            if (Swie<0) Strz=false;
            if (Skol>KOL || Skol<0 || Swie>WIE || Swie<0) {
                  Strz=false;
                  Skol=-2;
                  Swie=-2;
            }
```

```
        }
        }
        FKolo(Skol,Swie,"yellow");
}
```

FGlizdaStrzal *jeżeli można strzelać (jest ogon i nie wystrzelono wcześniej) to*
Strz=true *od tego momentu nie moa strzelać, aż pocisk wyleci za planszę*
IleOgo-- *zmniejszamy ogon o jeden*
ustalamy współrzędne i szybkość pocisku – takie jak głowa glizdy
rysujemy żółte koło pocisku
FObliczStrzal *sprawdzamy, co po drodze zestrzelił pocisk*
pocisk leci trzy razy szybciej niż automat glizdy, dlatego trzeba w pętli analizować trzy ruchu pocisku do przodu
kol != ”” *jeżeli pocisk trafił na jabłko*
switch *sprawdzamy co to za jabłko i dodajemy lub ujmujemy z ogona*
i wywołujemy funkcję FUstawOgon, która przestawia współrzędne ogona
zmieniamy współrzędne pocisku o jeden i sprawdzamy czy pocisk za ekranem
a jeżeli poza ekranem, to nie można strzelać i żeby go nie było widać na ekranie, to współrzędne (-2,-2)

- Strzel pociskiem i zjedz jabłko
- Wklej do ramki zrzut strony i notatnika z kodem JS



Automat – ruch (2)

Czy można grać z komputerowym „myślącym” przeciwnikiem?
W pierwszym etapie uruchomimy drugą automatyczną glizdę. W kolejnym nauczymy ją myśleć.

- Do dokumentu HTML, przed funkcją **FGlizda**, wpisz deklaracje zmiennych z ramki

```
var Akol=50;
var Awie=30;
var AkolV=-1;
var AwieV=0;
var IleAut=0;
var AUT=[];
```

zmienne odpowiedzialne za ruch automatycznej glizdy
nasza automatyczna glizda porusza się na razie cały czas w lewo

- Do dokumentu HTML, za etykietą **idOG**, wpisz znaczniki z ramki

```
Auto: <label id=idAU></label>
```

tutaj będzie widać długość ogobna automatycznej glizdy

- W dokumencie JS, w funkcji **FGlizda**, przed funkcją **clearTimeout**, wpisz polecenie z ramki

```
Automat();
```

wszystkie obliczenia związane z automatyczną glizdą w pętli głównej programy

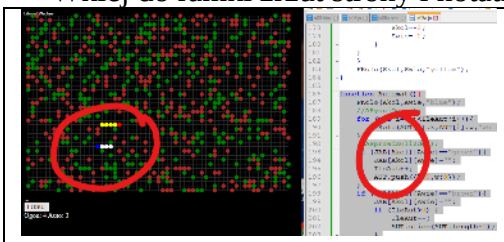
- Do dokumentu JS, wklej tekst z ramki

```
function Automat(){
      FKolo(Akol,Awie,"blue");
      //Rysuj Ogon
      for (var i=0;i<IleAut;i++){
            FKolo(AUT[i].k,AUT[i].w,"white")
      }
      //Sprawdz w co trafiła głowa
      if (JAB[Akol][Awie]=="green"){
            JAB[Akol][Awie]="";
            IleAut++;
            AUT.push({k:0,w:0});
      }
      if (JAB[Akol][Awie]=="brown"){
            JAB[Akol][Awie]="";
            if (IleAut>0) {
```

```
                  IleAut--;
                  AUT.slice(AUT.length-1);
            }
      }
      idAU.innerHTML=IleAut;
      //Prtzestaw Ogon
      for (var i=IleAut;i>=0;i--){
            AUT[i-1]=AUT[i-2];
      }
      AUT[0]={k:Akol,w:Awie};

      //ruch głowy
      Akol=Akol+AkolV;
      Awie=Awie+AwieV;
      //brzegi
      if (Akol>KOL) Akol=0;
      if (Akol<0) Akol=KOL;
      if (Awie>WIE) Awie=0;
      if (Awie<0) Awie=WIE;
}
```

Wszystkie niezbędne operacje w jednej funkcji
rysowana jest niebieska głowa automatu
rysujemy biały ogon
sprawdzamy, czy głowa nie zżarła jabłka
wpisujemy długość ogona do etykiety
przestawiamy tablicę ogona
obliczamy nowe współrzędne głowy
sprawdzamy przejścia przez brzeg planszy

- Umieść na planszy mnóstwo jabłek, aby automatyczna glizda miała ogon
- Wklej do ramki zrzut strony i notatnika z kodem JS



Glizda myśli (2)

Glizda szuka najbliższego jabłka - odległość do niego wylicza z tw. Pitagorasa
W dalszej kolejności sprawdza, który kierunek jest krótszy i ten wybiera, chyba, że już jest na tym samym pionie lub poziomie
Jeżeli zero jabłek na tablicy, to automat losuje kolejne

- W dokumencie JS, w funkcji **Automat**, przed instrukcjami wyliczającymi pozycję głowy, wpisz polecenie z ramki

```
AUstalKierunek();
```

wszystkie obliczenia związane z automatyczną glizdą w pętli głównej programy

- Do dokumentu JS, wklej tekst z ramki

```
function ASzukajJablka(Ak,Aw){
      var min={k:0,w:0,odl:999}
      for (var k=0;k<KOL;k++){
      for (var w=0;w<WIE;w++){
            if (JAB[k][w]=="green"){
                  var dk=Ak-k;
                  var dw=Aw-w;
                  var odl=Math.sqrt(dk*dk+dw*dw);
                  if (odl<min.odl) {
                        min.k=k;
                        min.w=w;
                        min.odl=odl;
                  }
            }
      }
```

```
	}
	return min;
}

function AUstalKierunek(){
	var min=ASzukajJablka(Akol,Awie);
	if (min.odl<999){
	var dk=Akol-min.k;
	var dw=Awie-min.w;
	if (Math.abs(dk) < Math.abs(dw)) {
		if (Math.abs(dk)>0 ){
			AkolV=Math.sign(-dk);AwieV=0;
		} else {
			AwieV=Math.sign(-dw);AkolV=0;
		}
	}
	else {
		if (Math.abs(dw)>0 ) {
			AwieV=Math.sign(-dw);AkolV=0;
		} else {
			AkolV=Math.sign(-dk);AwieV=0;
		}
	}
	} else {//gdy jest co szukac
		FUstawJablka();
	}
}
```

ASzukajJablka *szuka najbliższego jabłka*
min *w zmiennej zapisana pozycja i odległość do najbliższego jabłka od głowy glizdy*
w pętlach FOR szukamy jabłek i wyliczamy odległość do jabłka
jeżeli wyliczona odległość jest mniejsza niż zapisana w min, to zamieniamy
w efekcie po zakończeniu sprawdzania, w min będzie najbliższe jabłko
funkcja zwraca tablicę min
AUstalKierunek *w jakim kierunku ma iść glizda, żeby dojść do jabłka*
najpierw szukamy najbliższego jabłka za pomocą ASzukajJablko
wyliczamy odległości w pionie i poziomie do jabłka
wybieramy kierunek ten z mniejszą odległością
ale nie może być na tym samym poziomie (pionie)
a gdy nie ma czego szukać, bo brak jabłek, to losowanie nowych

- Spróbuj powalczyć z automatyczną glizdą i zbierz 20 jabłek w ogonie
- Wklej do ramki zrzut strony i notatnika z kodem JS