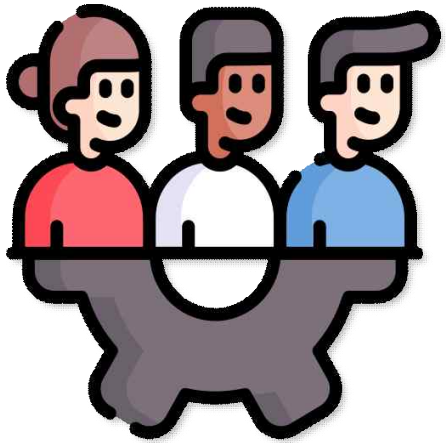
Game Programming Presentation

게임프로그래밍 발표

목차 List

팀원 소개



팀원

최원욱 박상민 한세윤

Development Plan

개발 계획

플래시365 처럼 여러 미니 게임을 한 곳에 모아
제공하는 웹 플랫폼을 모티브

기초 소스 탐색 및 개발 진행

- Clicker 게임

- Fishing 게임

- Runner 게임

Clicker Game

Clicker Game

기초 코드

```
function ClickerGame(){ this.targets=[]; this.time=30; this.score=0; this.over=false; this.spawn=0; }
ClickerGame.prototype.init = function(){ this.targets.length=0; this.time=30; this.score=0; this.over=false; this.spawn=0; };
ClickerGame.prototype.update = function(dt){ if(this.over) return; this.time -= dt; if(this.time<=0){ this.time=0; this.over=true; return; }
  this.spawn -= dt; if(this.spawn<=0){ this.spawn=E.rnd(0.4,0.8); this.targets.push({ x:E.rnd(40,E.width-40), y:E.rnd(100,E.height-80), r:E.rnd(16,26), life:E.rnd(1.2,2.2)}); }
  // shrink / expire
  for(const t of this.targets){ t.life -= dt; if(t.life<=0) t.remove=true; }
  // click
  if(E.mouse.down){ const m={x:E.mouse.x, y:E.mouse.y}; for(const t of this.targets){ const dx=m.x-t.x, dy=m.y-t.y; if(dx*dx+dy*dy <= t.r*t.r){ this.score += 5; t.remove=true; }} }
  this.targets = this.targets.filter(t=>!t.remove);
};
ClickerGame.prototype.draw = function(){ const c=E.ctx; E.clear('#fff7e8');
  // bg waves
  c.strokeStyle = '#f2c97a'; c.lineWidth = 2; for(let i=0;i<6;i++){ c.beginPath(); for(let x=0;x<=E.width;x+=10){ const y=120+i*40 + Math.sin((x+i*20)/40)*6; if(x===0) c.moveTo(x,y); else c.lineTo(x,y);} c.stroke(); }
  // targets
  for(const t of this.targets){ c.beginPath(); c.fillStyle='#6ed0ff'; c.arc(t.x,t.y,t.r,0,Math.PI*2); c.fill(); c.beginPath(); c.strokeStyle='#1f6d86'; c.lineWidth=2; c.arc(t.x,t.y,t.r+2,0,Math.PI*2); c.stroke(); }
  E.drawText('클리커: 목표를 빠르게 터치/클릭', 12, 8, '#5a4a16', 14);
};
ClickerGame.prototype.getScore = function(){ return this.score; };
Object.defineProperty(ClickerGame.prototype,'isOver',{ get(){ return this.over; }});
```

- 생성자 기반 상태 관리

time, score, target 등 저장

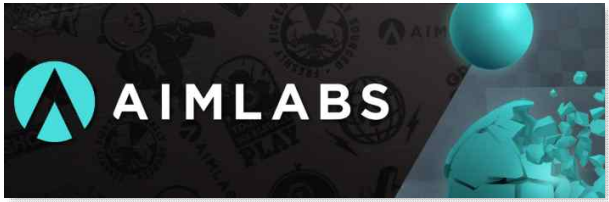
- update() 함수

객체 생성/삭제

충돌(클릭 판정)

Code Upgrade

코드 업그레이드



(Unity 기반의 대표적인 클릭 게임)

- 클릭 기반 게임 > 웹에서의 Aim traing 게임 개발

- 웹 에서의 3D 구현

- 시각 효과 업그레이드

- 게임성 확장

Code Upgrade

코드 업그레이드

babylon.js

- 바빌론 엔진 사용

- 카메라 설정

-3D 맵 모델링

-사운드

무료 소리 창고

Sketchfab

beretta_92_clean.glb

Code Upgrade

코드 업그레이드

Code Upgrade

코드 업그레이드

-총구 화염 , 탄피 배출

-조준선

- 격발 시 총구 pitch 회전 을 통해 반동 표현

- 적중 시 사운드 & 피격 효과 생성으로 타격감

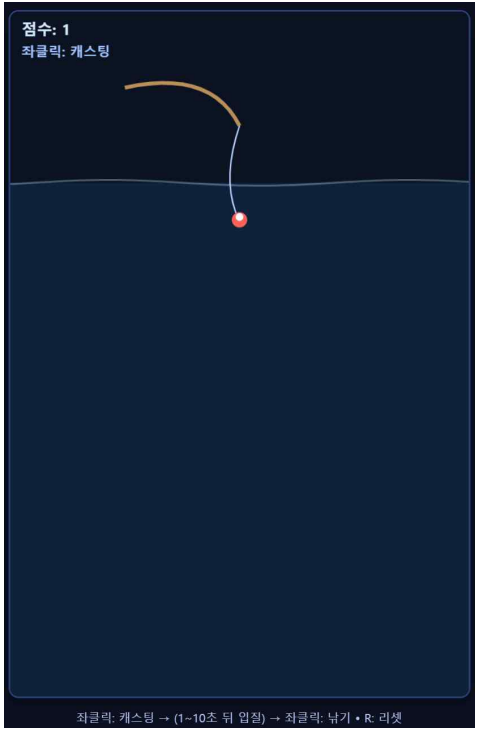
Fishing Game

소스코드의 변화

```
// 규칙: 좌클릭으로 캐스팅 → 1~10초 랜덤 후 '입질' → 좌클릭 성공(획득) / 놓치면 실패
(function(){
  const E = window.Engine;

  function SimpleFishing(){
    this.state = 'ready'; // ready|casted|bite|caught|miss
    this.timer = 0;       // 대기/입질 남은 시간
    this.score = 0;

    // 연출용
    this.rod = { x: E.width/2, y: 120 };
    this.bobber = { x: E.width/2, y: 220, bob: 0 };
    this.lineTo = { x: this.bobber.x, y: this.bobber.y };
  }

  SimpleFishing.prototype.reset = function(){
    this.state='ready';
    this.timer=0;
    this.bobber.x = E.width/2;
    this.bobber.y = 220;
    this.lineTo.x = this.bobber.x;
    this.lineTo.y = this.bobber.y;
  };

  SimpleFishing.prototype.update = function(dt){
    // 리셋 키
    if(E.keys.has('r')){ this.score=0; this.reset(); }

    // 클릭 에지
    const clicked = E.consumeClick();

    // 상태 머신
```

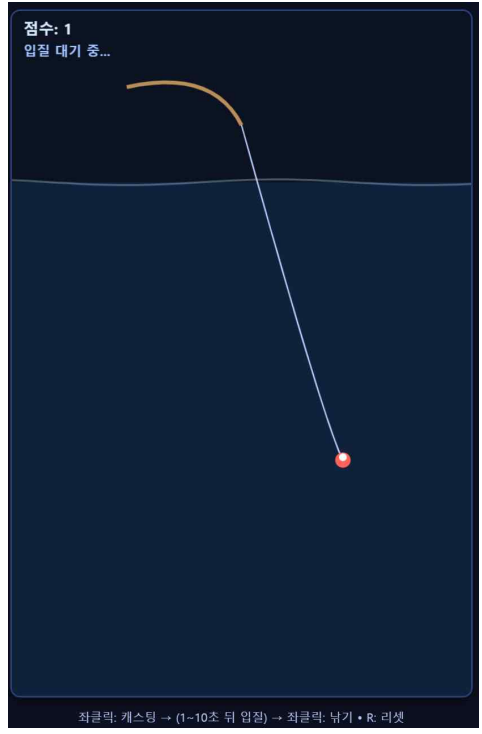
```
// ---------- Game ----------
function GameFishing(){
  this.state = 'ready';                              // ready → cast_timing → wait → mini
  this.score = 0;
  this.lastCatchText = '';

  this.waterTop = 180;
  this.lane = { x: E.width*0.5, top: this.waterTop+30, bottom: E.height-60 };

  // 플레이어 바(미니게임)
  this.bar = { y:0, h:120, v:0 };

  // 물고기
  this.fish = { y:0, vy:0, baseY:0, t:0, oscT:0, burstT:0, spd:120, profile:FISH_PROFILES.
  this.fishSize = 'small';
  this.zone = 'surface';

  // 진행도
  this.catchMeter = 0;
  this.grace = 0;

  // 타이밍 바(언더테일식)
  this.timing = { x:24, y:0, w:16, h:0, markerY:0, dir:1, speed:360, centerY:0, perfectWin

  // 찌 좌표
  this.castX = E.width*0.6;
  this.castY = this.waterTop+60;

  this.depth01 = 0.4;
  this.cfg = depthToConfig(this.depth01);

  this._resTimer = 0;
  this._waveT = 0;
```

기존 코드



기본화면

클릭 후 화면

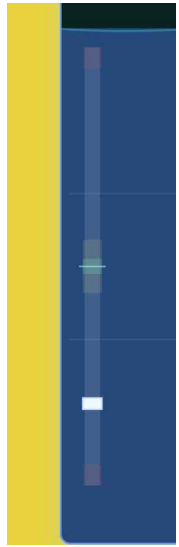


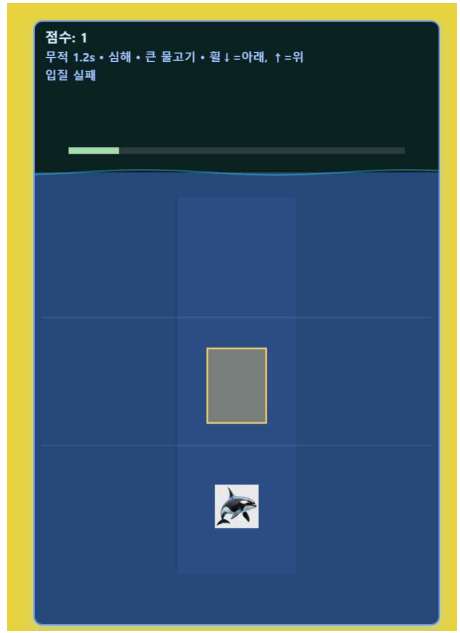
중요 코드 바뀐점

- 코드에 미니게임을 2개추가
  1. 찌를 던질때 미니게임 추가(언더테일 참고)
     - 바 사이에 목표물이 왔다갔다거리고 스페이스바를 이용해 중앙을 마추는 게임
  2. 물고기를 낚을때 미니게임 추가(스타듀밸리 참고)
     - 움직이는 물고기를 마우스 휠을 통해 계속 마추어 게이지를 누적시켜 물고기를 낚음

(언더테일의 미니게임)


(스타듀밸리의 미니게임)





1. 물고기의 상하 증폭 증가
   - 물고기의 크기별 '몸부림'의 강도가 심해짐(진폭, 주파수, 버스트 확률)
2. 물고기의 단계와 나오는 물고기의 위치 확률 변경
   - 수면/중앙/심해 구역, 캐스팅 정확도에 따라 구역 결정
3. 적절한 물고기 사진 적용
4. 게임 색상 적용 및 bgm, 소리 적용

캐스팅 '타이밍바' 미니게임

1. 시작 : _startCastTiming()
   - 수면 기준으로 타이밍 바의 위치(y)와 높이(h)를 계산
   - 바의 중앙선(centerY), 마커 초기 위치(markerY), 이동 방향(dir)을 세팅
   - 게임 상태를 cast_timing으로 바꿔 미니게임 진입
2. 루프 : state === 'cast_timing'
   - 매 프레임 markerY += dir * speed * dt로 마커를 위/아래로 이동
   - 위아래 경계(yMin, yMax)에 닿으면 방향 반전해서 왕복
   - 입력(스페이스/엔터/클릭) 발생 시 확정 함수 호출
3. 확정 : _confirmCastFromTiming()
   - 정확도 계산: dist = |markerY - centerY|
   - 깊이·구역 결정
     - depth = 1 - clamp(dist/0.5) 같은 방식으로 0~1 스케일을 만들고
     - zone = surface / mid / deep로 매핑
   - 물고기 크기/확률 설정: 정확도가 좋을수록 큰 물고기 비중이 높아짐

```
// 타이밍 바
if(this.state==='cast_timing'){
  const t = this.timing;
  t.markerY += t.dir * t.speed * dt;
  const yMin = t.y + 8, yMax = t.y + t.h - 8;
  if(t.markerY <= yMin){ t.markerY = yMin; t.dir = +1; }
  if(t.markerY >= yMax){ t.markerY = yMax; t.dir = -1; }
  if(E.keys.has(' ') || E.keys.has('enter') || clicked){
    this._confirmCastFromTiming();
  }
  return;
}
```

```
// --- 타이밍 바 시작 ---
GameFishing.prototype._startCastTiming = function(){
  const y = this.waterTop+20;
  const h = E.height - this.waterTop - 80;
  Object.assign(this.timing, { y, h, centerY: y + h*0.5, markerY: y + 8, dir: 1 });
  this.state = 'cast_timing';
};

// --- 타이밍 확정 → zone/size/depth/찌 위치 결정 ---
GameFishing.prototype._confirmCastFromTiming = function(){
  const t = this.timing;
  const dist = Math.abs(t.markerY - t.centerY);
```

낚시 미니게임

1. 진입(상태전환)
   - cast_timing에서 _confirmCastFromTiming()으로 zone/depth/size와 난이도 cfg를 생성
   - 짧은 wait 후 this.state = 'minigame'
   - 초기화: this.fill=0~1, this.bar={y,v,h}, this.fish={y,t,profile}, this.timeSinceStart=0
2. 입력 &바 물리(휠 조작)
   - 휠 아래=위로 / 휠 위=아래로(반전) 가속
   - 가속·속도제한·감속·경계처리
   - 바의 높이(판정폭) = this.bar.h
3. 물고기 AI
   - const F = this.fish.profile
   - 목표 궤적: 사인파 + 지터 + 버스트
   - 추종(부드럽게 따라감): this.fish.y += (target - this.fish.y) * (F.follow * dt)
   - 범위 고정: ln.top..ln.bottom
4. 겹침 판정 > 캐치 게이지
   - 바 중앙: barC = this.bar.y + this.bar.h/2
   - 겹침 여부: inside = |this.fish.y - barC| ≤ this.bar.h/2
   - 누적: this.fill = clamp(this.fill + delta*dt, 0, 1)
5. 성공/실패 판정&점수
   - this.fill ≥ 1 → this.state='caught'
   - this.fill ≤ 0 → this.state='miss'
   - 점수 가중: 물고기 크기별 small=1, mid=3, big=5 추가
   - 단축키: R로 즉시 리셋(테스트용)

```
// 미니게임
if(this.state==='minigame'){
  const ln=this.lane;

  // 바 이동(휠 내림=아래, 올림=위)
  if(Math.abs(wheel)>0){
    const notch = (wheel>0)? +1 : -1;
    this.bar.v += notch * this.cfg.wheelStep;
    this.bar.v = clamp(this.bar.v, -this.cfg.barVMax, this.cfg.barVMax);
  }else{
    this.bar.v *= Math.exp(-this.cfg.friction*dt);
    if(Math.abs(this.bar.v)<5) this.bar.v = 0;
  }
  this.bar.y += this.bar.v*dt;
  if(this.bar.y < ln.top){ this.bar.y = ln.top; this.bar.v = 0; }
  if(this.bar.y > ln.bottom - this.bar.h){ this.bar.y = ln.bottom - this.bar.h;
    this.bar.v = 0; }
```

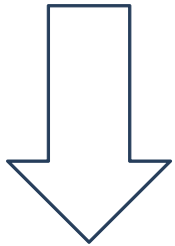
```
// 상태별
if(this.state==='ready'){
  E.drawText('스페이스/엔터/클릭 → 캐스팅 타이밍!', 12, 36, '■#9dc1ff', 14);
  this.drawCastGaugeFrame();
}else if(this.state==='cast_timing'){
  E.drawText('왼쪽 세로바: 중앙에 맞춰 놓려라! (중앙 가까울수록 심해 확률↑)', 12, 36, '■#9c
  this.drawCastTimingBar();
}else if(this.state==='wait'){
  E.drawText('입질 대기 중...', 12, 36, '■#9dc1ff', 14);
  this.drawBobber(this.castX, this.castY);
  this.drawCastGaugeFrame();
}else if(this.state==='minigame'){
  this.drawFishingMini();
}else if(this.state==='caught' || this.state==='miss'){
  E.drawText(this.state==='caught'?'잡았다!':'놓쳤다...', 12, 36, '■#9dc1ff', 14);
  this.drawBobber(this.castX, this.castY);
  this.drawCastGaugeFrame();
}
```

Runner Game

Runner

러너 - 소스코드

미니게임에 들어갈 쿠키런 스타일의 러너 게임을 만들고 싶은데,
그 기반이 될 기초 소스코드를 작성해 주세요.



1. 플레이어 기본 설정
2. 물리 엔진 요소
3. 게임 상태 변수
4. 입력처리

```
(function(){
  const E = window.Engine;

  function RunnerGame(){
    this.player = { x: 80, y: 600, w: 32, h: 42, vy: 0, onGround: true };
    this.groundY = 640;
    this.gravity = 1600;
    this.jumpV = -420;
    this.speed = 240;
    this.time = 0;
    this.over = false;
  }

  RunnerGame.prototype.init = function(){
    this.time = 0;
    this.over = false;
    this.player.y = this.groundY - this.player.h;
    this.player.vy = 0;
    this.player.onGround = true;
  };

  RunnerGame.prototype.control = function(){
    const jumpHeld = E.keys.has(' ') || E.keys.has('arrowup');
    if (jumpHeld && this.player.onGround){
      this.player.vy = this.jumpV;
      this.player.onGround = false;
    }
  };

  RunnerGame.prototype.update = function(dt){
    if(this.over) return;
    this.time += dt;
    this.control();
    this.player.vy += this.gravity * dt;
    this.player.y += this.player.vy * dt;
    if(this.player.y + this.player.h >= this.groundY){
      this.player.y = this.groundY - this.player.h;
      this.player.vy = 0;
      this.player.onGround = true;
    }
  };

  RunnerGame.prototype.draw = function(){
    const c = E.ctx;
    E.clear();
    c.fillStyle = '#f3d96d';
    c.fillRect(0, this.groundY, E.width, E.height - this.groundY);
    E.drawRoundRect(this.player.x, this.player.y, this.player.w, this.player.h, 8,
  };

  window.RunnerGame = RunnerGame;
})();
```

Runner

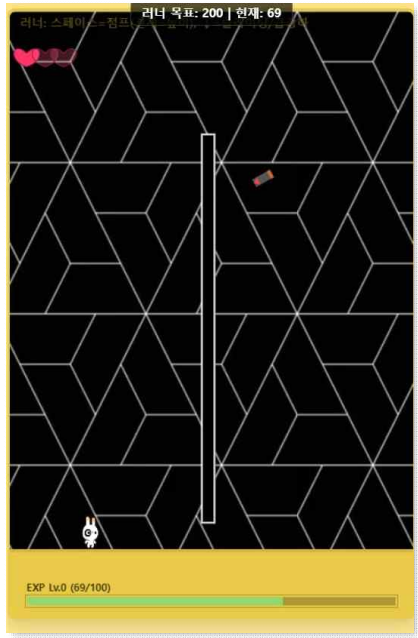
러너 - 기획

초반
- ✓ 진행 중에 부딛힐 장애물 필요
- ✓ 점프 뿐만 아니라 슬라이딩, 급강하 구현
- ✓ 주인공/장애물/배경 이미지, 배경음악, 효과음 필요

중반
- ✓ 점수 대신 레벨업 시스템 구현
- ✓ 추가 장애물 미사일과 러너 속도 점진적 증가
- ✓ 배경 이미지가 움직이지 않는걸 방지하기 위해 반복되는 이미지로 교체
- ✓ 목숨 3개 구현

후반
- ✓ 카드를 고른 후 카운트 다운 적용
- ✓ 버그픽스

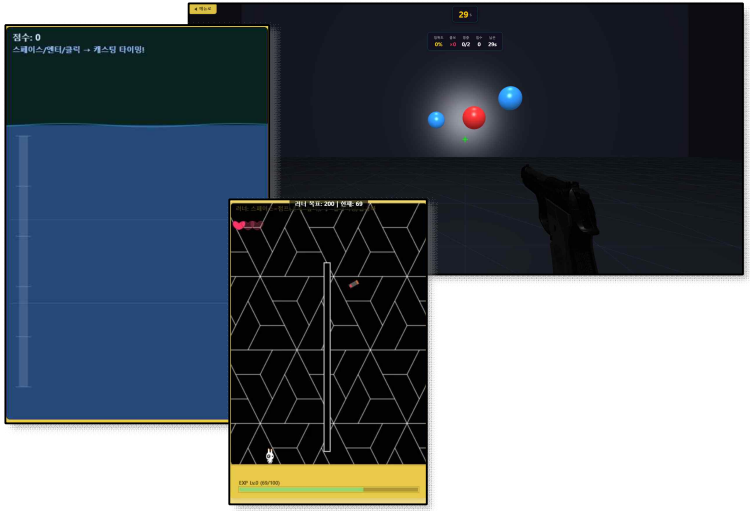
FlappyBird에서 영감을 받음



Challenge

챌린지

3개의 게임을 병합한 메인 게임




```
(function(){
  // 챌린지 모드 설정
  const config = {
    rounds: 5,
    timeLimitFishing: 30,
    timeLimitClicker: 30,
    thresholds: {
      // 러너 스코어 목표
      runner:  [200, 500, 800, 1100, 1500],
      // 낚시 스코어 목표
      fishing: [ 2, 4, 6,  8,  10],
      // 에임트레이너 클릭 수 목표
      clicker: [300, 450, 600,  800, 1000]
    }
  };

  function shuffle(arr){ const a = arr.slice(); for(let i=a.length-1;i>0;i--)
    {const j=(Math.random()* (i+1))|0; [a[i],a[j]]=[a[j],a[i]];} return a; }

  window.Challenge = {
    config,
    shuffle,
    getTargets(roundIndex){
      return {
        runner:  config.thresholds.runner[roundIndex],
        fishing: config.thresholds.fishing[roundIndex],
        clicker: config.thresholds.clicker[roundIndex],
        timeFishing: config.timeLimitFishing,
        timeClicker: config.timeLimitClicker
      };
    }
  };
})();
```

게임 시연