models.py - Board

```
def validate_col_range(value):
    if value < 0 or value >= MAX_COLS:
        raise ValidationError('Column out of range', code='col_value')

def validate_row_range(value):
    if value < 1 or value >= MAX_ROWS:
        raise ValidationError('Row out of range', code='row_value')

class Board(models.Model):
    label = models.CharField(max_length=1)
    row = models.IntegerField(validators=[validate_row_range])
    col = models.IntegerField(validators=[validate_col_range])
    value = models.IntegerField()

    @classmethod
    def create_board(cls, row, col):
        model = cls(label=BOARD_STR, row=row, col=col, value=0)  # does not validate
        return model

    def __str__(self):
        return f'{self.label} @({self.row}, {self.col}) ${self.value}'
```

models.py - Player

```
class Player(models.Model):
    name = models.CharField(max_length=1)
    row = models.IntegerField(validators=[validate_row_range])
    col = models.IntegerField(validators=[validate_col_range])
    score = models.IntegerField()

    @classmethod
    def create_player(cls, name, row, col):
        model = cls(name=name, row=row, col=col, score=0)  # does not validate
        return model

    def __str__(self):
        return f'{self.name} @({self.row}, {self.col}) ${self.score}'
```

urls.py

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
    path('create/', views.create, name='create'),
    path('display/<str:name>/', views.display, name='display'),
    path('move/<str:name>/', views.move, name='move'),

]
```

views.py

```
from django.db import transaction
from django.db.models import Q
from django.http import HttpResponseRedirect, Http404
from django.shortcuts import render
from random import randrange
from .constants import MAX_COLS, MAX_ROWS, NUM_TREASURES, MIN_VALUE, MAX_VALUE, NUM_PLAYERS, BOARD_STR,
TREASURE_STR
from .models import Board, Player

def index(request):
    return display(request, '0')

@transaction.atomic
def create(request):
    Board.objects.select_for_update().all().delete()
    Player.objects.select_for_update().all().delete()

    for i in range(MAX_ROWS):
        for j in range(MAX_COLS):
            # print(f'Creating {i} {j}')
            b = Board.create_board(i, j)
            b.save()
```

Must lock rows; will be unlocked when the transaction ends

views.py

```
for _ in range(NUM_TREASURES):
while True:
    x = randrange(MAX_ROWS)
    y = randrange(MAX_COLS)
    b = Board.objects.select_for_update().filter(Q(row=x), Q(col=y))[0]
    if b.value == 0:
        b.value = randrange(MIN_VALUE, MAX_VALUE + 1)
        b.save()
        break

for n in range(NUM_PLAYERS):
    while True:
        x = randrange(MAX_ROWS)
        y = randrange(MAX_COLS)
        b = Board.objects.select_for_update().filter(Q(row=x), Q(col=y))[0]
        p = Player.objects.select_for_update().filter(Q(row=x), Q(col=y))
        if b.value == 0 and len(p) != 1:
            name = str(n + 1)
            b.label = name
            b.save()
            p = Player.create_player(name, x, y)
```

views.py

```
                p.save()
                break

    # print('Done.')
    return HttpResponseRedirect(redirect_to='/game/display/0')

def display(request, name):
    board_set = Board.objects.all().order_by('row', 'col')

    ctr = 0
    board = []
    row = []
    for b in board_set:
        row.append(b)
        ctr += 1
        if ctr % MAX_COLS == 0:
            board.append(row)
            row = []

    context = {'board': board, 'name': name, 'players': Player.objects.all().order_by('name')}

    return render(request, 'game/board.html', context)
```

views.py

```
@transaction.atomic
def move(request, name):
    try:
    cmd = request.POST['button_id']
    p = Player.objects.select_for_update().filter(name=name)  # presumably sanitized by Django...
    if len(p) != 1:
        raise Http404(f'Due to unknown name')

    new_row = p[0].row
    new_col = p[0].col
    match cmd:
        case "up":
            new_row -= 1
        case "down":
            new_row += 1
        case "left":
            new_col -= 1
        case "right":
            new_col += 1
        case _:
            raise Http404(f'Due to unknown direction')

    if new_row < 0 or new_col < 0 or MAX_ROWS <= new_row or MAX_COLS <= new_col:
        return HttpResponseRedirect(redirect_to=f'/game/display/{name}')
```

views.py

```
b = Board.objects.select_for_update().filter(Q(row=p[0].row), Q(col=p[0].col))[0]
new_b = Board.objects.select_for_update().filter(Q(row=new_row), Q(col=new_col))[0]
if new_b.label != BOARD_STR and new_b.label != TREASURE_STR:
    return HttpResponseRedirect(redirect_to=f'/game/display/{name}')

p[0].row = new_row
p[0].col = new_col
b.label = BOARD_STR
new_b.label = name
if new_b.value > 0:
    p[0].score += new_b.value
    new_b.value = 0
new_b.save()
b.save()
p[0].save()
except Exception as details:
    raise Http404(f'Due to: {details}')

return HttpResponseRedirect(redirect_to=f'/game/display/{name}')
```

Exception is too broad; will also catch *DatabaseError*, counteracting *@transaction.*atomic